LUCIANA BOLAN FRIGO

MathTutor
UM AMBIENTE INTERATIVO MULTIAGENTE PARA O ENSINO DE ESTRUTURA DA INFORMAÇÃO

FLORIANÓPOLIS
2002

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

MathTutor
UM AMBIENTE INTERATIVO MULTIAGENTE PARA O ENSINO DE ESTRUTURA DA INFORMAÇÃO

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para
obtenção do grau de Mestre em Engenharia Elétrica

LUCIANA BOLAN FRIGO

Florianópolis, março de 2002.

MathTutor

UM AMBIENTE INTERATIVO MULTIAGENTE PARA O ENSINO DE ESTRUTURA DA INFORMAÇÃO

Luciana Bolan Frigo

Esta dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia Elétrica, Área de Concentração em Controle, Automação e Informática Industrial, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.

Guilherme Bittencourt, Dr.
Orientador

Edson de Pieri, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Guilherme Bittencourt, Dr
Presidente

Luiz Fernando Jacintho Maia, Dr.

Marcelo Stemmer, Dr.

Ricardo Rabelo, Dr.

Agradecimentos

Ao meu orientador, Guilherme Bittencourt pela oportunidade de crescimento pessoal e profissional, além da orientação.

À Capes pelo apoio financeiro, Universidade Federal de Santa Catarina, CNPq e LCMI pela estrutura fornecida.

Aos participantes do projeto Math_net, Antônio, Carlos,Guilherme, Gustavo, Leonardo e Paulo que colaboraram para este trabalho, em especial a Terezinha de Fátima Faria.

Aos meus pais, Luiz e Cirlene, por toda dedicação e esforço para que tudo isto tornasse possível. Ao meu noivo, Cleicio, pelo apoio e paciência. Ao meu irmão, Tiago e todos os familiares que vibram com minhas conquistas.

A todos os amigos e colegas que direta ou indiretamente ajudaram na realização deste projeto.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica.

MathTutor

UM AMBIENTE INTERATIVO MULTIAGENTE PARA O ENSINO DE ESTRUTURA DA INFORMAÇÃO

Luciana Bolan Frigo

Março/2002

Orientador: Guilherme Bittencourt, Dr.

Área de Concentração: Controle, Automação e Informática Industrial

Palavras-chave: Inteligência Artificial Aplicada a Educação, Ambientes Interativos de Aprendizagem

Número de Páginas: 86

Desde os anos 60, o uso do computador como ferramenta auxiliar de ensino vem incorporando novas tecnologias e criando questionamentos que só podem encontrar resposta na multidisciplinaridade, na combinação de elementos de psicologia, ergonomia, pedagogia, além de diversas áreas da ciência da computação. Com o surgimento da Internet, a questão do ensino por computador universalizou-se, levando muitas instituições a investir em ambientes computacionais para ensino-aprendizagem. O presente trabalho apresenta o desenvolvimento de um Sistema Tutor Inteligente (STI) chamado MathTutor que visa facilitar o aprendizado dos principais conceitos de abstração de dados e de procedimentos em linguagens de programação. O MathTutor está baseado em um modelo para concepção e desenvolvimento de ambientes de aprendizagem assistidos por computador, chamado MATHEMA. Visando a implementação dos diversos componentes exigidos pelo modelo MATHEMA, foram utilizadas diversas ferramentas computacionais. A portabilidade levou à escolha da linguagem de programação Java e à utilização do protocolo HTML e Servlets como mecanismos de interface. Utilizou-se ainda as seguintes ferramentas de domínio público: JATLite, um sistema para implementação de agentes, o arcabouço de sistemas especialistas Jess, o interpretador de Lisp CLISP e o sistema de banco de dados PostgreSQL. A principal contribuição deste trabalho é a integração destes elementos em uma arquitetura com as funcionalidades definidas no modelo MATHEMA, possibilitando a construção de cursos de acordo com a metodologia proposta neste modelo.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

MathTutor

A MULTI-AGENT INTERACTIVE ENVIROMENT TO TEACH INFORMATION STRUCTURE

Luciana Bolan Frigo

March/2002

Advisor: Guilherme Bittencourt, Dr.

Area of Concentration: Control, Automation and Industrial Computing

Keywords: Artificial Intelligence in Education, Learning Interactive Environment.

Number of Pages: 86

Since the sixties, the use of the computer as an educational tool is continuously incorporating new technologies and generating doubts and questions. Any answer to these questions must be a multidisciplinary one, combining elements from psychology, ergonomy, pedagogy, besides several areas of computer science. The Internet universalized the discussion about computer-based education and lead many institutions to invest in computational environments to improve teaching processes. This work presents the development of an intelligent tutoring system (ITS) called MathTutor. The system aims to facilitate the learning of the main concepts about procedural and data abstractions of programming languages. The MathTutor system is based on a model for conception and development of computer-based learning environments, called MATHEMA. In order to implement the different components of the MATHEMA model, several computational tools were used. Portability concerns lead to the choice of the Java programming language and to the use of the HTML protocol and Servlets as interface mechanisms. The following public domain tools were also used: JATLite, an agent implementation system, the expert system shell Jess, the CLISP Lisp interpreter and the PostgreSQL data base system. The main contribution of this work is the integration of these elements into an architecture with the functionalities defined in the MATHEMA model, allowing to build courses using the metodology's model.

Sumário

1 INTRODUÇÃO 1

1.1 O SISTEMA MATHTUTOR 2

1.2 ESTRUTURA DA DISSERTAÇÃO 4

2 INTELIGÊNCIA ARTIFICIAL E EDUCAÇÃO 5

2.1 INTRODUÇÃO 5

2.2 INFORMÁTICA NA EDUCAÇÃO 5

2.3 SISTEMAS ESPECIALISTAS 6

2.3.1 Arquitetura de um SE 7

2.3.2 Representação do Conhecimento 8

2.3.3 Métodos de Representação de Conhecimento 9

2.4 SISTEMAS TUTORES INTELIGENTES 10

2.4.1 Arquitetura de um STI 11

2.5 CONCLUSÃO 14

3 SISTEMAS MULTIAGENTES 15

3.1 INTRODUÇÃO 15

3.2 DEFINIÇÕES DE AGENTES 15

3.3 CLASSIFICAÇÃO DOS AGENTES 16

3.4 DEFINIÇÃO DE UM SISTEMA MULTIAGENTE 18

3.5 ARQUITETURA DE SMA 19

3.5.1 Arquitetura Quadro-Negro 19

3.5.2 Arquitetura de Troca de Mensagens 19

3.5.3 Arquitetura Federativa 20

3.6 CONCLUSÃO 20

4 MATHEMA 21

4.1 INTRODUÇÃO 21

4.2 ARQUITETURA GERAL DO MODELO MATHEMA 21

4.3 MODELAGEM DO CONHECIMENTO SOBRE UM DOMÍNIO 23

4.3.1 Interações no MATHEMA 25

4.4 SOCIEDADE DE AGENTES TUTORES ARTIFICIAIS 26
4.4.1 Modelo de agente tutor *26*
4.4.2 Arquitetura de um Sistema Tutor *29*
4.5 CONCLUSÃO 31

5 MATHTUTOR 32

5.1 INTRODUÇÃO 32
5.2 QUADRO COMPARATIVO 32
5.2.1 Servidor de Páginas Web *33*
5.2.2 Sistema de Arquivos *33*
5.2.3 Interface *33*
5.2.4 Agentes *34*
5.2.5 Disponibilidade *34*
5.3 FERRAMENTAS 34
5.3.1 Suporte de Inteligência Artificial *36*
5.3.2 Suporte de comunicação de agentes *38*
5.3.3 Suporte multimídia *39*
5.4 ARQUITETURA DO MATHTUTOR 42
5.5 INTERAÇÃO COM O USUÁRIO 46
5.6 COMUNICAÇÃO 53
5.7 CONFIGURAÇÃO E ORGANIZAÇÃO 55
5.8 TRABALHOS RELACIONADOS 57
5.9 CONCLUSÃO 59

6 CONCLUSÕES E PERSPECTIVAS 60

6.1 CONTRIBUIÇÕES 61

APÊNDICE A 62

APÊNDICE B 67

APÊNDICE C 72

APÊNDICE D 75

Lista de Figuras

Figura 2-1 Arquitetura de um Sistema Especialista 8
Figura 4-1 Arquitetura Geral do MATHEMA 22
Figura 4-2 Interações no MATHEMA 25
Figura 4-3 Agente Tutor no Nível Macro 27
Figura 4-4 Agente Tutor no Nível Micro 28
Figura 5-1 Interface com Resultado de uma Busca 37
Figura 5-2 Camadas do JATLite 38
Figura 5-3 Processo de Servidor para executar Servlets 41
Figura 5-4 Funções do Especialista 42
Figura 5-5 Arquitetura do MathTutor 43
Figura 5-6 Interface 45
Figura 5-7 Interface Antiga 46
Figura 5-8 Página Resultante da Inserção de Conteúdo 48
Figura 5-9 Página de Cadastramento no MathTutor 49
Figura 5-10 Login do MathTutor 49
Figura 5-11 Página para Efetuar Anotações 50
Figura 5-12 Interpretação de um código 50
Figura 5-13 Visualização dos dados pessoais cadastrados 51
Figura 5-14 Área Exclusiva para Categoria Professor 51
Figura 5-15 Esquema da comunicação entre os principais componentes do sistema 53
Figura 5-16 Comunicação entre os agentes do MathTutor 54
Figura 5-17 Configuração básica do mapa de arquivos do sistema 55
Figura 5-18 Troca de Mensagem entre os Agentes 56
Figura 5-19 AulaNet 57
Figura 5-20 ELM-ART Lisp Course 58
Figura 5-21 Eletrotutor 59
Apêndice A
Figura A 1Modelo de Agente 64
Apêndice B
Figura B 1 Diagrama de Caso de Uso 67

Figura B 2 Caso de Uso 68
Figura B 3 Diagrama de Seqüência de Login 68
Figura B 4 Diagrama de Seqüência do Cadastro 69
Figura B 5 Professor Solicita Alunos Cadastrados 69
Figura B 6 Aprendiz Solicita Conteúdo 70
Figura B 7 Aprendiz Faz Anotações 70
Figura B 8 Interpretação de um Exercício 71
Figura B 9 Aprendiz Verifica suas Anotações 71

1 Introdução

Desde os anos 60, o uso do computador como ferramenta auxiliar de ensino vem incorporando novas tecnologias e criando questionamentos (o que ensinar? como ensinar? a quem ensinar?) que só podem encontrar resposta na multidisciplinaridade, na combinação de elementos de psicologia, ergonomia, pedagogia, além de diversas áreas da ciência da computação. Com o surgimento da Internet, a questão do ensino por computador universalizou-se, levando muitas instituições a investir em ambientes computacionais com o objetivo de melhorar o processo de ensino-aprendizagem e colaborar com a democratização do conhecimento. Algumas das promessas desta nova tecnologia são:

(1). gerar uma forma mais atraente de ensino, que se adapte às necessidades de cada usuário, onde este avança no curso de acordo com o seu tempo de assimilação e disponibilidade, podendo rever determinado assunto quantas vezes forem necessárias;

(2). democratizar o ensino atendendo àqueles que residem em locais onde não há instituições de ensino convencionais, ou que estão dispersos geograficamente;

(3). incentivar a educação permanente, satisfazendo a crescente demanda das aspirações dos mais diversos grupos com a promoção de atividades de extensão educacional e cultural;

(4). promover um ensino inovador e de qualidade, pela sua sistemática e recursos didáticos instrucionais e de multimídia, além da comunicação bidirecional como garantia para uma aprendizagem dinâmica e inovadora [30].

Dentre as tecnologias utilizadas para aprimorar os Sistemas Tutores Inteligentes (STI's), a Inteligência Artificial (IA) em geral e a Inteligência Artificial Distribuída (IAD) em particular foram de grande valia, dando origem a área atualmente conhecida com Inteligência Artificial Na Educação (AI-ED). A IAD é uma das áreas da IA que mais se desenvolveram nos últimos anos e apresenta um enorme potencial de aplicações, estuda o conhecimento e as técnicas de raciocínio que podem ser necessárias ou úteis para que agentes computacionais participem de sociedades de agentes.

1.1 O Sistema MathTutor

O presente trabalho apresenta o desenvolvimento de um Sistema Tutor Inteligente (STI) chamado MathTutor que visa facilitar o aprendizado dos principais conceitos de abstração de dados e de procedimentos em linguagens de programação. Estes conceitos fazem parte do conteúdo da disciplina de Fundamentos da Estrutura da Informação, aplicada no curso de Engenharia de Controle e Automação Industrial na Universidade Federal de Santa Catarina. O MathTutor é um sistema distribuído, utilizando técnicas de Inteligência Artificial Distribuída (IAD), seguindo a abordagem de Sistemas Multiagentes (SMA). MathTutor começou a ser desenvolvido na dissertação de mestrado de Terezinha de Fátima Faria [23] e esta dissertação apresenta as diferenças entre o trabalho realizado por Terezinha e o estado atual do projeto.

O MathTutor está baseado no MATHEMA [15] que é um modelo para concepção e desenvolvimento de ambientes de aprendizagem assistidos por computador e faz parte do projeto Math_net -- Uma abordagem via sistemas multiagentes para concepção e realização de ambientes interativos de aprendizagem cooperativa assistidos por computador [33] -- um projeto de Pesquisa em Informática na Educação apoiado pelo CNPQ-ProTeM- CC, cujo tema central é a concepção e o desenvolvimento de um modelo computacional para Ambientes Interativos de Ensino-Aprendizagem Cooperativa com base em múltiplos agentes artificiais e humanos. O MATHEMA foi concebido em uma tese de doutorado, onde o principal objetivo é uma metodologia para o desenvolvimento de Sistemas Tutores Inteligentes (STI).

Visando a implementação dos diversos componentes exigidos pelo modelo MATHEMA, foram utilizadas diversas ferramentas computacionais. Inicialmente, o modelo MATHEMA, como qualquer outro modelo de STI, exige uma interface adequada. A portabilidade levou à escolha da linguagem de programação Java e à utilização do protocolo HTML e Servlets [33] como mecanismos de interface.

A base do modelo MATHEMA é a chamada Sociedade de Agentes Tutores Artificiais (SATA), cuja implementação exige uma ferramenta que suporte uma sociedade de agentes cognitivos, visto que cada agentes é um STI completo, apenas com seu domínio delimitado de acordo com a metodologia de estruturação do domínio sugerida pelo modelo MATHEMA. A implementação dos agentes que compõem o sistema é suportada pelo sistema JATLite [32], que permite a criação de agentes de software e sua comunicação

através do protocolo KQML. A capacidade cognitiva dos agentes é suportada pelo arcabouço de sistemas especialistas Jess [25].

O modelo MATHEMA incorpora uma metodologia de organização e apresentação do domínio a ser ensinado. Esta metodologia é baseada na resolução de problemas. Cada lição do MathTutor está dividida em três partes: teoria, exemplos e exercícios, exigindo que os STI presentes nos agentes da SATA tenham uma certa capacidade de resolução de problemas. No caso do curso de Fundamentos da Estrutura sa Informação, os problemas são em geral expressos em linguagem de programação Lisp, o que leva à utilização de um interpretador Lisp como solucionador de problemas. O interpretador utilizado foi o CLISP [13], uma implementação do padrão Common Lisp [55].

A adaptação ao aprendiz, no modelo MATHEMA, é realizada através da construção de um modelo do aprendiz. Este modelo, que contém desde informações cadastrais até todo o histórico do aluno ao longo de suas seções no sistema, é armazenado em um sistema de banco de dados. O sistema utilizado foi o PostgreSQL [46]. Todas as ferramentas utilizadas são de domínio público.

A principal contribuição deste trabalho é a integração destes elementos em uma arquitetura com as funcionalidades definidas no modelo MATHEMA, possibilitando a construção de cursos de acordo com a metodologia proposta neste modelo.

As principais etapas deste trabalho foram:

1. Criar os agentes e fazê-los comunicarem-se;

2. Implementar para cada agente o sistema especialista baseado em regras responsável por seu comportamento;

3. Desenvolver uma interface amigável padronizada, que permita a interação do aprendiz com o MathTutor;

4. Integrar todos os itens mencionados anteriormente de acordo com o modelo MATHEMA [12];

5. Disponibilizar o sistema via Internet.

Além das etapas citadas anteriormente uma outra funcionalidade importante a ser mencionada e adicionada no decorrer do projeto é a monitoração das ações do aprendiz, através do uso de agentes, com a ação pedagógica que melhor se adapta ao aprendiz.

Outros trabalhos semelhantes estão sendo desenvolvidos como o Shart-Web, um STI para o ensino de música [3].

1.2 Estrutura da Dissertação

O trabalho encontra-se organizado em 6 capítulos.

No Capítulo 2 ("Inteligência Artificial e Educação") apresentam-se noções de Inteligência Artificial, apresentando os Sistemas Especialistas (SE) e os Sistemas Tutores Inteligentes (STI), além da relação existente entre Inteligência Artificial e Educação.

No Capítulo 3 ("Sistemas Multiagentes") são abordados os principais conceitos e definições deste campo de pesquisa.

No Capítulo 4 ("MATHEMA") são apresentadas noções sobre projetos instrucionais, associadas às características do MATHEMA e ao MathTutor. A seguir, no Capítulo 5 ("MathTutor"), são abordadas as etapas de desenvolvimento do MathTutor, apresentando as metas atingidas, as ferramentas utilizadas, entre outros aspectos não menos importantes no desenvolvimento do sistema.

No Capítulo 6 ("Conclusões e Perspectivas") faz-se uma análise global do trabalho realizado. Discutem-se os principais benefícios, expectativas, e desafios encontrados durante a execução do mesmo.

No Apêndice A apresenta-se um paralelo entre noções de agentes sob o ponto de vista da Inteligência Artificial e Engenharia de Software.

No Apêndice B estão alguns diagramas escolhidos para facilitar a compreensão do funcionamento do MathTutor.

E finalmente, no Apêndice C mostra como a modelagem do conhecimento do domínio e no Apêndice D o código de programação de um dos agentes.

2 Inteligência Artificial e Educação

2.1 Introdução

A integração da Inteligência Artificial com os Ambientes de Aprendizagem Assistidos por Computador originou a área de pesquisa denominada Inteligência Artificial em Educação (IA-ED).

Neste capítulo serão abordados os principais aspectos relacionados a este tema que formam o embasamento teórico desta dissertação.

No contexto da pesquisa de IA-ED considera-se a questão da aprendizagem como atividade fundamental em Educação. Diversos modelos pedagógicos podem ser seguidos no tratamento da questão da aprendizagem, sendo estes divididos nas seguintes abordagens:

(1). Behaviorista: baseada em um modelo estímulo-resposta. Tem perdido espaço, diferentemente da abordagem cognitivista.

(2). Cognitivista: situa-se na linha construtivista, considerando o conhecimento como o resultado de um processo de construção pessoal.

(3). Situacionista: considera que a aprendizagem está localizada num contexto e é permanentemente evolutiva, sendo um processo de iniciação, integração e colaboração no interior de diferentes comunidades de prática.

Do ponto de vista da implementação computacional, desde a década de 90 até a atualidade, destaca-se a tendência para modelos baseados na cooperação, ocorrendo a evolução dos Sistemas Tutores Inteligentes tradicionais para Ambientes Interativos/Inteligentes de Aprendizagem.

2.2 Informática na Educação

Nos anos 70, o ensino apoiado por computador prometia mudanças profundas nas formas de se conceber e administrar o processo de ensino-aprendizagem [49]. Estas promessas levaram a uma mudança nos conceitos de aprendizagem, provocando uma revolução quando se percebeu que o computador poderia auxiliar no processo de aprendizagem, criando uma nova mídia educacional [60].

É necessário saber que esta nova mídia educacional não substitui o professor, mas possibilita uma mudança dos papéis deste e do seu aluno. O professor passaria de transmissor de conhecimento para criador de ambientes de aprendizagem e facilitador.

O termo *Informática na Educação* possui diversos significados dependendo da visão educacional e pedagógica em que o computador é utilizado. Segundo a visão do grupo que desenvolveu o sistema MathTutor, *Informática na Educação* significa o uso do computador no processo de aprendizagem dos conteúdos curriculares.

Existem duas maneiras de se utilizar o computador na educação. Uma seria através da extensão do processo instrucionista, onde o computador continua transmitindo a informação para o aluno. Esta é mais simples de ser implementada e executada. A outra, cria condições para que o aluno construa seu próprio conhecimento através de ambientes de aprendizagem que façam uso do computador.

A primeira abordagem é a aplicada no ensino tradicional, onde o computador é utilizado apenas para informatizar os processos de ensino existentes. Esta abordagem apesar de ser a mais utilizada, é bastante questionável no mundo moderno, onde o mercado de trabalho busca profissionais dinâmicos, criativos e capazes de construírem seu próprio conhecimento.

O uso do computador na criação de ambientes de aprendizagem que enfatizam a construção do conhecimento apresenta enormes desafios, redimensionando conceitos, idéias e valores já conhecidos. O processo de formação deve oferecer condições para o professor construir conhecimento sobre técnicas computacionais e entender porque e como integrar o computador na sua prática pedagógica.

Ambientes inteligentes de aprendizagem por computador possuem sistemas especialistas que tentam imitar o comportamento do professor em uma determinada situação, facilitando o aprendizado do aluno em um determinado conteúdo.

2.3 Sistemas Especialistas

Sistemas Especialistas (SE), uma aplicação da inteligência artificial, são sistemas que utilizam o conhecimento e o representam explícita e separadamente do módulo principal do sistema [56].

São tradicionalmente vistos como sistemas de suporte à decisão, pois são capazes de tomar decisões como um especialista em uma área específica. A parte mais sensível no desenvolvimento de um Sistema Especialista é a aquisição do conhecimento [9].

2.3.1 Arquitetura de um SE

Um dos princípios fundamentais no projeto dos SE é a separação do conhecimento de domínio com o código do sistema.

A arquitetura de um SE contém uma base de regras, uma memória de trabalho e um motor de inferência [9]. A base de regras e a memória de trabalho formam a chamada base de conhecimento do SE, onde está representado o conhecimento sobre o domínio. Este conhecimento precisa ser organizado de uma maneira adequada para que o motor de inferência consiga tratá-lo convenientemente. O conhecimento em um sistema especialista consiste de fatos e heurística. Os fatos constituem as informações que estarão sempre disponíveis para serem compartilhadas pelo especialista do domínio. As heurísticas são regras práticas que caracterizam o nível da tomada de decisão do especialista em um domínio. Uma base de conhecimento pode ser vista como um conjunto de regras, cada qual podendo ser validada independentemente da estrutura de controle. Um dos problemas mais críticos na implementação de SE é fornecer um conhecimento completo sobre o qual o sistema vai operar. O nível de desempenho de um sistema especialista está relacionado ao tamanho e a qualidade de sua base de conhecimento.

O motor de inferência é o mecanismo de controle do sistema que avalia e aplica as regras de acordo com as informações da memória de trabalho. O motor de inferência tenta imitar os tipos de pensamento que o especialista humano emprega quando resolve um problema, ou seja, ele pode começar com uma conclusão e procurar uma evidência que a comprove, ou pode iniciar com uma evidência para chegar a uma conclusão. Estes dois métodos são chamados de regressivo (do inglês *backward chaining)* e progressivo *(*do inglês *forward chaining)*, respectivamente. Nem todos os sistemas utilizam a mesma abordagem para a representação do seu conhecimento, portanto, o motor de inferência deve ser projetado para trabalhar com a representação de conhecimento específica utilizada.

A interface facilita a comunicação entre o SE e o usuário, permitindo apresentar ao aluno perguntas e explicações através da entrada de fatos e dados.

Sistemas Especialistas estão sendo bastante utilizados na área médica, onde estes permitem agilizar e facilitar o atendimento ao paciente. O sistema especialista mais freqüentemente citado é o MYCIN [53], desenvolvido por uma equipe de médicos e especialistas em IA na Universidade de Stanford. O MYCIN contém conhecimento especializado no campo de doenças infecciosas. Foi projetado para auxiliar no diagnóstico e tratamento de meningite (inflamação das membranas que envolvem o cérebro e a medula espinhal) e bacteriemia (infecção bacteriana no sangue). O MYCIN utiliza o tipo de raciocínio regressivo. Dando-se um conjunto de sintomas para diagnóstico, o MYCIN utiliza seus conhecimentos para determinar suas causas e então recomendar o tratamento apropriado. O MYCIN foi um pioneiro entre os sistemas especialistas e representa um esforço de aproximadamente 50 homens/ano. Muito deste esforço está embutido em sua base de conhecimento, com suas 450 regras.

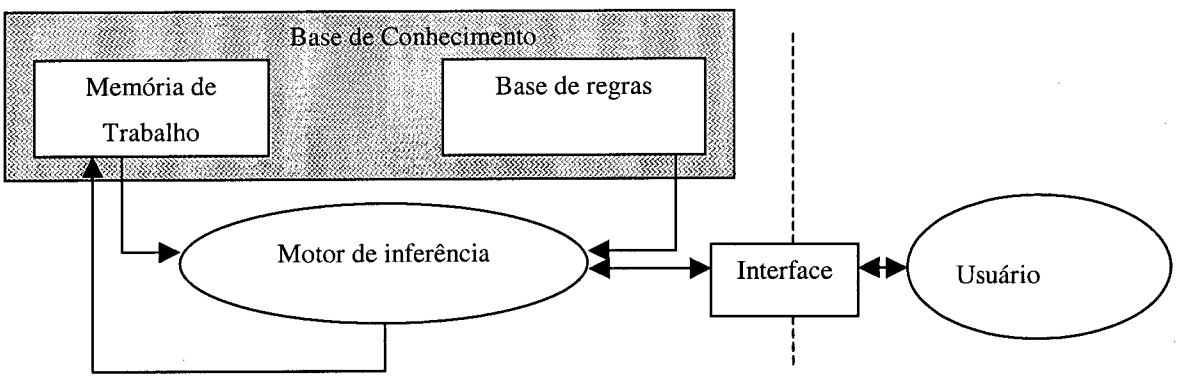

Figura 2-1 Arquitetura de um Sistema Especialista

2.3.2 Representação do Conhecimento

Todo programa de computador contém o conhecimento sobre um determinado problema a ser resolvido. O conhecimento está nos algoritmos que o programa emprega e nos procedimentos de decisão que determinam quais destes algoritmos empregar em determinada circunstância.

A linha básica de pesquisas seguida pela IA ao longo dos anos tem sido a de simular inteligência através de programas de computador cuja característica principal é a utilização de conhecimento através de sua representação explícita [62].

Para se resolver os complexos problemas encontrados em Inteligência Artificial, precisa-se tanto de uma grande base de conhecimento, como de mecanismos para manipular este conhecimento e criar soluções para novos problemas.

Para representar fatos é necessário algum tipo de formalismo [48]. A parte mais importante no projeto de um SE é a escolha do método de representação de conhecimento. Em tese, uma representação geral como a lógica seria suficientemente expressiva para representar qualquer tipo de conhecimento. No entanto, problemas de eficiência, facilidade de uso e a necessidade de expressar conhecimento incerto e incompleto levaram ao desenvolvimento de diversos tipos de formalismos de representação de conhecimento [9].

2.3.3 Métodos de Representação de Conhecimento

Entre os diversos tipos de formalismos para representar o conhecimento, mencionamos: Regras de produção, Lógica, Redes Semânticas e Quadros.

1. Regras de produção: São compostas por duas partes, uma condição e uma conseqüência, por exemplo, "se este animal tem escamas, não é mamífero". Uma das vantagens em se usar regras é que são modulares, ou seja, encapsulam conhecimento e podem ser facilmente expandidas [48].

2. Lógica: é a base para a maioria dos formalismos de representação de conhecimento. Trabalha com a manipulação de variáveis lógicas que representam proposições. As proposições podem ser verdadeiras ou falsas e podem ser combinadas através de conectivos. Mesmo os formalismos não lógicos têm, em geral, seu significado formal descrito através de uma especificação lógica de seu comportamento.

3. Redes Semânticas: uma rede semântica consiste em um conjunto de nodos conectados por um conjunto de arcos. Os nodos em geral representam objetos e os arcos, relações binárias entre esses objetos. Mas os nodos podem também ser utilizados para representar predicados, classes, palavras de uma linguagem, entre outras possíveis interpretações, dependendo do sistema de redes semânticas em questão [9].

4. Quadros: um quadro é basicamente uma coleção de atributos e valores associados que descrevem alguma entidade no mundo. Foram introduzidos para permitir a expressão das estruturas internas dos objetos, mantendo a possibilidade de representar herança de propriedades como as redes semânticas.

As idéias fundamentais destes último método foram introduzidas por Marvin Minsky [39] em seu artigo *A framework to represent knowledge.* As aplicações propostas por Minsky para o novo método foram análise de cenas, modelagem da percepção visual e compreensão de linguagem natural [9].

No caso do MathTutor foram utilizadas as regras de produção, pois o conhecimento a ser representado é pouco estruturado, além de ser baseado na prática.

2.4 Sistemas Tutores Inteligentes

A maioria dos SE não foram projetados para ensinar, tendo pouca utilidade direta do ponto de vista educacional, principalmente devido à ausência de qualquer estratégia educacional, incapacidade de comparar o conhecimento do estudante em relação ao do especialista e a incapacidade de determinar a ação quando o conhecimento do estudante difere do conhecimento do especialista [57]. Entretanto, a estrutura do sistema especialista serve perfeitamente para ser adaptada para a construção de sistemas tutoriais, proporcionando um grande potencial para a criação de novos ambientes educacionais.

Um Sistema Tutor inteligente (STI) é definido, segundo Auberger [4], como um sistema instrucional baseado em computador, que ensina o estudante de maneira interativa, usando os conceitos de Inteligência Artificial.

Murray [41] ressalta como um dos objetivos dos STIs ser capaz de modelar complexos comportamentos de ensino, os quais se adaptam às necessidades do estudante, à situação de aprendizagem e ao assunto da instrução.

O sistema GUIDON [12] é um sistema tutorial inteligente para o ensino de diagnóstico de doenças infecciosas do sangue, que foi desenvolvido a partir da base de conhecimento já formada do MYCIN [53], ou seja, é um sistema especialista adaptado ao ensino.

2.4.1 Arquitetura de um STI

As funções operacionais básicas de um STI são determinadas por quatro componentes principais ou modelos:

1. Modelo Especialista

Segundo Auberger [4], este módulo contém o conhecimento do domínio do sistema, e os mecanismos de inferência.

Este modelo serve como o conhecimento a ser apresentado e como um padrão para avaliar a performance do estudante.

O modelo do especialista é fundamentalmente uma base de conhecimento, contendo informações sobre um determinado domínio, que é organizada de alguma maneira para representar o conhecimento de um especialista ou professor. É, geralmente, considerado o componente central de qualquer STI. Em essência, este modelo incorpora a maior parte da "inteligência" do sistema na forma do conhecimento necessário para solucionar problemas do domínio [44].

Várias abordagens para modelar o especialista e representar seu conhecimento têm sido exploradas nas pesquisas em STIs. A aquisição do conhecimento necessária para este modelo pode ser a principal tarefa no desenvolvimento do sistema, requerendo muitas horas de colaboração entre o projetista e o especialista ou professor. Um modelo do especialista bem projetado facilitará a comunicação do conhecimento entre o professor e o estudante. Entretanto, quanto maior esta capacidade, maior será a complexidade do sistema, e isto, na maioria das vezes, pode não ser necessário para uma instrução eficaz. Pode ser suficiente representar este conhecimento para um conjunto reduzido de problemas a serem usados com propósitos educacionais, simplificando o desenvolvimento do modelo.

O grande desafio para cada novo STI é fornecer uma representação do seu domínio, rica o suficiente para suportar o nível desejado de compreensão proporcionando uma maior flexibilidade no ensino.

2. Modelo do Estudante

Este módulo descreve o conhecimento e crenças do estudante. É, na verdade, uma representação abstrata do estudante no sistema. A principal função do modelo do estudante é permitir que o conteúdo instrucional seja adaptado para um aprendiz individual. Com

conhecimento sobre o estudante, o tutor pode controlar a ordem e a dificuldade do material a ser apresentado, bem como prover remediações [4].

A chave para um ensino personalizado e inteligente em um sistema tutorial é, sem dúvida, o conhecimento que o sistema deve ter de seu próprio usuário. A dimensão mais significativa de um STI é sua capacidade para modelar o conhecimento do estudante [34].

A característica principal deste modelo deve contemplar todos os aspectos do conhecimento e do comportamento do estudante que tragam conseqüências para o seu desempenho e aprendizagem. Entretanto, a construção de um modelo como este é uma tarefa bastante complexa para um sistema computadorizado. Os canais de comunicação em um computador podem parecer bastante restritos quando comparados com a capacidade das pessoas em combinar informações em uma grande variedade de meios, como por exemplo o tom de voz ou expressões faciais.

Mesmo que para obter decisões pedagógicas razoáveis não se tenha a necessidade de construir um modelo completo do estudante ao longo de todas as suas dimensões, a construção de um modelo parcial que forneça somente as informações necessárias é, ainda hoje, um desafio para os projetistas de sistemas computacionais.

3. Modelo Pedagógico

Também conhecido como tutor ou instrucional, usa as informações do modelo do estudante para determinar quais aspectos do conhecimento do domínio devem ser apresentados para o Aprendiz. Representa os métodos e técnicas didáticas utilizadas no processo da comunicação de conhecimento.

Este modelo contém o conhecimento necessário para tomar decisões sobre quais táticas de ensino devem ser empregadas dentre aquelas disponíveis no sistema. As decisões e ações deste modelo são altamente dependentes dos resultados do processo de diagnóstico. O modelo pedagógico diagnostica as necessidades de aprendizagem do estudante com base nas informações do modelo do estudante e na solução do professor contida no modelo do especialista. Em geral, as decisões são sobre qual informação apresentar ao estudante, quando e como apresentá-la.

As decisões pedagógicas são tomadas no contexto de um ambiente educacional que determina o grau de controle sobre a atividade e sobre a interação possuídos respectivamente pelo sistema tutorial e pelo estudante [65].

Portanto, um processo de aprendizagem depende de uma grande variedade de fatores e o sistema tutorial deve cuidar para não destruir a motivação pessoal do estudante ou o seu senso de descobrimento. Este processo pedagógico requer grande versatilidade.

4. Modelo da Interface

É responsável por processar o fluxo de informações com o meio externo e o sistema. Traduz a representação interna do sistema numa linguagem compreensível para o aprendiz.

Na engenharia de software, a interface do usuário tem sido a primeira preocupação dos projetistas quando estão discutindo a criação de uma nova aplicação, pois como afirmam Hix e Hartson [29].

"Para os usuários, a interface é o próprio sistema".

Muitos princípios baseados em teorias cognitivas têm sido propostos para projetos de interface como resultado de pesquisas na área da interação homem-máquina. Entretanto, a meta da maioria destas pesquisas é que o usuário não necessite se adaptar à interface do sistema e sim, a interface deve ser projetada para que seja intuitiva e natural para ele aprender a utilizá-la.

Apesar da interface operar em estreita cooperação com ambos os modelos, pedagógico e do estudante, suas decisões são de natureza distinta, requerendo um tipo diferente de conhecimento.

Um outro aspecto relacionado com interfaces, e tradicionalmente relacionado com as pesquisas em IA, é o processamento da linguagem natural. Este ramo de pesquisa ainda é um grande obstáculo a ser vencido.

É importante salientar que em uma interação com o STI, o estudante não irá somente aprender o conteúdo das lições, mas também terá que aprender como utilizar o sistema, portanto, a facilidade de uso deve ser uma das considerações principais no projeto destas interfaces. Uma interface consistente, ajudará a reduzir a carga cognitiva sobre o estudante [52].

2.5 Conclusão

O MathTutor é um sistema instrucional baseado em computador, portanto, é um STI. Suas componentes principais estão baseadas no modelo MATHEMA [15].

Podemos então, definir alguns aspectos do sistema desenvolvido de acordo com o que foi apresentado neste capítulo como:

1. Dentre os métodos de representação de conhecimento o utilizado foram as regras de produção, pois são adequadas para representar conhecimentos pouco estruturados e baseados na prática.
2. Em consonância com as abordagens existentes para a questão da aprendizagem o MathTutor, de acordo com o modelo MATHEMA, segue a abordagem cognitivista.

No próximo capítulo apresenta-se uma visão geral sobre agentes e sistemas multiagentes.

3 Sistemas Multiagentes

3.1 Introdução

Os sistemas Multiagentes são um tipo de sistema inteligente que fazem parte da subárea da Inteligência Artificial chamada Inteligência Artificial Distribuída (IAD). A IAD diz respeito ao estudo e concepção de sistemas com várias entidades interagindo, logicamente e espacialmente distribuídas e que podem ser autônomas e inteligentes [64].

A IAD pode ser subdividida em Sistemas Multiagentes (SMA) e Solução Distribuída de Problemas (SDP).

Tanto nos SMA como na SDP os agentes compartilham um mesmo ambiente. A diferença está mais na forma de interação entre os agentes. Nos modelos SDP, os agentes são projetados de maneira que haja o máximo de cooperação. Nos modelos SMA, os agentes competem por recursos e precisam muitas vezes resolver conflitos [51].

3.2 Definições de Agentes

Pesquisadores que trabalham com agentes apresentam uma variedade de definições, cada uma tenta explicar a palavra *agente*.

Segundo Franklin & Graesser [24] um agente autônomo é um sistema inserido em um ambiente ou em parte dele que percebe este ambiente e age nele.

Agentes devem possuir as seguintes características segundo Wooldridge & Jennings [66]:

- Autonomia: operam sem a intervenção humana direta e possuem um controle sobre suas ações e seu estado interno.
- Sociabilidade: Interagem entre si e com os humanos através de uma linguagem comum.
- Reatividade: Agentes percebem o ambiente e reagem às mudanças que ocorrem nele.
- Pró-atividade: Os agentes não respondem simplesmente ao ambiente, eles são capazes de apresentar iniciativa própria.

Para Franklin & Graesser [24] um agente deve possuir quatro propriedades que são: Reatividade, Autonomia, Pró-atividade e devem ser Contínuos no tempo.

3.3 Classificação dos Agentes

A noção de agente autônomo tem um papel central na pesquisa contemporânea em Inteligência Artificial [17]. Um agente autônomo é um sistema com capacidade de decisão sobre as metas que orientarão sua atividade em um dado ambiente. Ele deve ser capaz de aprender com a experiência através de sua interação com o ambiente que o cerca, e generalizar esta aprendizagem para aprender melhor no futuro.

Pode-se distinguir duas classes principais de agentes autônomos: os assim chamados cognitivos (ou deliberativos, ou ainda simbólicos), e os reativos.

Os agentes cognitivos possuem um processo explícito para escolher a ação a ser tomada. Baseiam-se em mecanismos de processamento simbólico existentes nos sistemas mais tradicionais da Inteligência Artificial, como redes semânticas, sistemas de regras, etc.

Os agentes reativos possuem um processo de tomada de decisão algorítmico, em geral em tempo real, em resposta a estímulos do ambiente, usualmente captados por seus sensores. A escolha da ação está diretamente relacionada com a ocorrência de um conjunto de eventos que ele percebe no e do ambiente. Podem ser baseados em mecanismos computacionais alternativos como redes neurais, sistemas classificadores, processamento analógico, etc.

É importante observar que os sistemas multiagentes fazem parte de uma área interdisciplinar que envolve psicologia cognitiva e informática, entre outras. Os sistemas Multiagentess também podem ser divididos em sistemas reativos, cognitivos ou mistos, de acordo com os tipos dos agentes que formam o sistema.

Os sistemas de agentes reativos são constituídos por um grande número de agentes simples e fortemente acoplados. Não possuem inteligência ou representação de seu ambiente e interagem utilizando um comportamento do tipo estímulo/resposta [18]. Um comportamento inteligente emerge a partir das interações entre estes agentes e seu ambiente [40]. Sendo assim, os agentes não são inteligentes individualmente, mas o comportamento global é.

Os sistemas de agentes cognitivos são geralmente constituídos de um pequeno número (tipicamente menor que 50) de agentes, onde cada agente possui uma quantidade considerável de recursos. Estes agentes são inteligentes, contêm uma representação parcial e explícita de seu ambiente, têm uma capacidade local de decisão e podem negociar uma informação ou um serviço. Eles são em geral dotados de conhecimentos, competências,

intenções e planos, o que lhes permite coordenar suas ações visando a resolução de um problema. Seu comportamento é especificado declarativamente, isto é, o programador diz o que o agente deve ou não fazer. Possuem uma representação explícita do conhecimento e seu mecanismo de controle é deliberativo, ou seja, o agente raciocina sobre que ações realizar.

Nesta última classe, deve-se notar que cada agente trata unicamente o subproblema que requer especificamente sua competência. Eles podem desenvolver atividades cooperativas e coordenadas para resolver um problema. Desta forma, um sistema Multiagente pode ser definido em função da autonomia de cada agente e dos meios que o mesmo dispõem para gerenciar suas interações.

De modo geral, os agentes buscam simplesmente satisfazer seus objetivos individuais e a interação entre eles passa a existir somente quando surgem alguns conflitos, devido à falta de recursos locais ou simplesmente a necessidade de articulação de ações individuais. A ausência de controle global e de dados é compensada pelos mecanismos locais de coordenação definidos sobre o modelo dos outros agentes, de seus objetivos, de suas intenções e sobre os procedimentos de cooperação.

Segundo Ekdahl [20], os agentes reativos têm mostrado suas limitações, embora se mostrem mais eficientes do que os agentes cognitivos para uma série de tarefas relacionadas ao acesso direto ao mundo real. Além de não serem muito versáteis, eles tem problemas para obter conhecimento que dependa de raciocínio, pois o raciocínio envolve elementos que não estão diretamente ligados à percepção. Por outro lado, os agentes deliberativos têm dificuldade em construir conhecimento diretamente a partir do mundo real.

Além dos agentes puramente cognitivos ou reativos, poderia se pensar em outra abordagem, que seria o uso de agentes autônomos baseados em mecanismos híbridos. Esta abordagem não tem sido muito explorada [17], e a questão de como proceder para combinar os mecanismos cognitivo e reativo ainda merece muita pesquisa. Muitos pesquisadores têm sugerido que a síntese dos benefícios dos agentes reativos e dos cognitivos seria muito interessante para a pesquisa em agentes autônomos, uma vez que o sistema reativo poderia tratar as tarefas de rotina e o sistema deliberativo poderia ser usado para tarefas mais avançadas. Um exemplo deste tipo de abordagem é o UFSC-Team [59].

A utilização de estados mentais para a modelagem de agentes cognitivos é chamada de abordagem mentalista [61]. A idéia principal desta abordagem mentalista, segundo Corrêa

[14], se concentra no fato de que o agente cognitivo possui estado interno que se relaciona com o estado do ambiente com o qual interage. Estes estados seriam correspondentes aos estados mentais humanos, que apresentam um vínculo com o mundo em termos da sua existência e significância.

Crença, desejo, expectativa e intenção são exemplos de estados intencionais [27].

Uma crença de um agente corresponde às informações que o agente tem sobre o mundo (é o conhecimento do ambiente de forma explícita). Crenças podem ser vistas como simples variáveis (como por exemplo, na linguagem PASCAL), mas agentes modelados na arquitetura BDI[1] representam crenças de forma simbólica (como por exemplo, fatos em PROLOG) [68].

Um desejo de agente (ou objetivo em um sistema) intuitivamente corresponde a tarefas estabelecidas pelo próprio agente. Agentes BDI, assim como agentes humanos, não exigem que desejos sejam logicamente consistentes. Desejo é um estado mental intencional e com potencial motivador das ações do agente [68].

A intuição em sistemas BDI é que o agente não é capaz, geralmente, de realizar todos os seus desejos, mesmo sendo consistentes. Esses desejos escolhidos são adotados como intenções e um agente continuará a tentar realizar uma intenção até que acredite que a intenção foi satisfeita, ou acredite que a intenção não poderá ser realizada [68].

3.4 Definição de um Sistema Multiagente

Várias definições têm sido propostas para o termo *Sistema Multiagente*. Do ponto de vista da Inteligência Artificial Distribuída, um Sistema Multiagente é uma rede acoplada de entidades de resolução de problemas que trabalham juntas para achar respostas aos problemas que estão além das capacidades individuais ou do conhecimento de cada entidade [19].

Para a formação de uma sociedade de agentes visando solucionar um problema torna-se necessária a consideração de alguns conceitos como:

- Organização: um grupo de agentes, objetivando resolver um determinado problema, onde a idéia é realizar uma decomposição em subproblemas, repartidos entre os agentes envolvidos.

[1] BDI (do inglês Beliefs, Desires and Intentions)

- Controle: centralizado ou distribuído
- Cooperação: mecanismos que viabilizam as atividades cooperativas entre agentes.
- Comunicação: mecanismos responsáveis pela comunicação entre agentes, normalmente baseados na troca de mensagens ou compartilhamento de informações.

3.5 Arquitetura de SMA

Algumas abordagens [63] de arquiteturas de sociedades de agentes são apresentadas a seguir.

3.5.1 Arquitetura Quadro-Negro

Em sociedades baseadas na arquitetura quadro-negro, os agentes, chamados fontes de conhecimento, não se comunicam diretamente, mas sempre através do quadro-negro. Quadro-negro é uma estrutura de dados, geralmente persistente, onde existe uma divisão em regiões ou níveis, visando facilitar a busca de informações. Nesta arquitetura, todas as interações entre os agentes se dão através dessa estrutura onde os mesmos escrevem e lêem suas mensagens quando necessário. Pode-se assim dizer que um quadro-negro é uma memória de compartilhamento global onde existe uma quantidade de informações e conhecimento usados para leitura e escrita pelos agentes. A medida que o número de agentes aumenta, o processo de gravação e recuperação das mensagens no quadro-negro pelos agentes pode se tornar demorado demais para um sistema de tempo real.

3.5.2 Arquitetura de Troca de Mensagens

Na arquitetura de troca de mensagens, os agentes comunicam-se diretamente enviando mensagens assíncronas. Não existe o papel do quadro-negro como intermediário do processo de interação, mas pode haver a presença de um agente facilitador de comunicação.

Esse tipo de arquitetura exige que os agentes saibam os nomes e endereços uns dos outros para que as mensagens sejam devidamente encaminhadas. Esse método é mais eficiente no sentido de obter as mensagens em tempo hábil.

Para que as trocas de mensagens ocorram de maneira adequada entre os agentes é necessário estabelecer um protocolo de conversação. O protocolo é quem dita as regras e

impõe o formalismo necessário para que as mensagens sejam encaminhadas e compreendidas pelos agentes.

3.5.3 Arquitetura Federativa

Considerando um sistema de troca de mensagens onde o número de agentes é muito grande, a emissão de uma mensagem em difusão (*broadcasting*) levará um tempo muito alto podendo até inviabilizar todo processo de comunicação do sistema.

Diante desse problema, surgiu a arquitetura federativa onde os agentes da sociedade são divididos em grupos ou federações segundo um critério de agrupamento escolhido. Junto a cada grupo de agentes encontram-se os agentes facilitadores responsáveis por receber a mensagem que chega em cada grupo e encaminhá-la para o agente destinatário presente naquele grupo. A vantagem dessa arquitetura é que o agente facilitador tem a propriedade de identificar se a mensagem que chega é destinada a algum agente de seu grupo e se for o caso, fazer a devida entrega.

A arquitetura federativa propõe a diminuição do fluxo de mensagens desnecessárias entre os agentes que formam a sociedade, pois os facilitadores têm a capacidade de remetê-las ao respectivo destinatário sem a necessidade de enviá-las a todos os agentes.

3.6 Conclusão

Neste capítulo descrevemos a subárea da IA conhecida como SMA. Neste contexto, o MathTutor é um Sistema Multiagente cognitivo pois seus agentes são cognitivos, isto é, estão baseado em um mecanismo de processamento simbólico, como sistemas de regras, descritos no capítulo anterior. Os agentes no MathTutor estão organizados em domínios e subdomínios (Apêndice C) que indicam a responsabilidade de cada agente em um determinado domínio do sistema, tendo assim um controle distribuído.

A arquitetura do MathTutor funciona com a troca de mensagens, que é explicada no Capítulo 5, pois este mecanismo apresenta as mensagens em tempo hábil e não torna o sistema demasiadamente lento no caso de muitas mensagens serem trocadas num intervalo pequeno de tempo, como ocorreria na arquitetura quadro-negro.

4 MATHEMA

4.1 Introdução

O MathTutor, foco deste trabalho, foi baseado no modelo MATHEMA, proposto no âmbito da tese de doutorado de Evandro de Barros Costa [15].

O MATHEMA é um modelo de ambiente interativo de aprendizagem, concebido para possibilitar um ensino adaptativo e seus desdobramentos. O ensino adaptativo é visto como conseqüência do processo de interações cooperativas envolvendo os seus componentes (Aprendiz, Tutor) em atividades baseadas em resolução de problemas. O sistema envolve o aprendiz na resolução de problemas e passa a desempenhar um papel de tutor assistente.

Será feita uma breve apresentação dos aspectos fundamentais do modelo MATHEMA para o entendimento deste trabalho.

4.2 Arquitetura Geral do Modelo MATHEMA

Definiu-se uma sociedade de agentes tutores especializados em operar sobre o conhecimento distribuído neste modelo e a concepção de um modelo baseado numa arquitetura Multiagente, considerando principalmente as atividades de interação cooperativa envolvendo três entidades: um *aprendiz humano (AH)*, uma *sociedade de agentes tutores artificiais (SATA)* e uma *sociedade de especialistas humanos (SEH)*.

A arquitetura geral do MATHEMA pode ser visualizada na Figura 4-1:

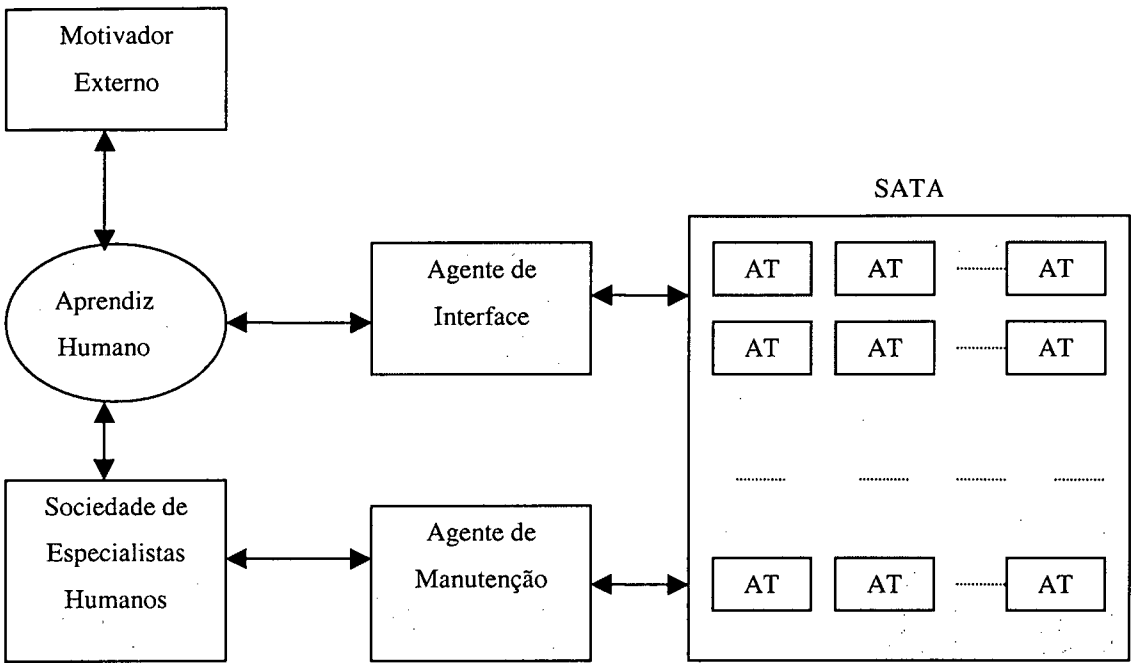

Figura 4-1 Arquitetura Geral do MATHEMA

Onde:

Aprendiz Humano (AH): agente interessado em aprender o conteúdo sobre um dado domínio.

Sociedade de Agentes Tutores Artificiais (SATA): conjunto de agentes capazes de cooperar entre si a fim de promover a aprendizagem de um dado aprendiz. Cada agente integrante da SATA é especializado em subdomínios relacionados a um dado domínio de conhecimento. Essa idéia foi inicialmente inspirada nas reflexões contidas em *Sociedade da Mente[40].*

Sociedade de Especialistas Humanos (SEH): funciona como uma fonte de conhecimento externa ao sistema e se comunica com a SATA através dos agentes de manutenção. Suas principais funções são promover o processo de manutenção da SATA, que diz respeito à inclusão e exclusão de agentes, bem como alterações no conhecimento dos agentes e estar disponível para que, em casos onde a SATA não consiga resolver um determinado problema, possa auxiliar os agentes e assistir os aprendizes.

Agente de Interface (AI): é o agente responsável pela interação entre o aprendiz e a SATA.

Agente de Manutenção(AM): é o agente responsável pela interação entre a SATA e a SEH. Sua função é a de prover esta interação, oferecendo meios necessários para a percepção, comunicação e manutenção da SATA.

Motivador Externo (ME): representa as entidades humanas externas que podem motivar o aprendiz à utilização do MATHEMA.

Portanto:

$$Arq - Mat = \{AH, SATA, SEH, AI, AM, ME\} \qquad 4\text{-}1$$

4.3 Modelagem do conhecimento sobre um domínio

A técnica utilizada para a modelagem do domínio no MATHEMA foi concebida com o intuito de atenuar a complexidade inerente ao processo ensino-aprendizagem.

No ambiente MATHEMA a modelagem do conhecimento sobre um dado domínio é abordada segundo duas dimensões: uma visão externa e uma visão interna.

A visão externa mostra o conhecimento sob um aspecto de diferentes subdomínios para o domínio alvo. O particionamento é de natureza epistemológica, estando comprometido com alguma visão particular do domínio. Visando alcançar um conhecimento com especialidades distribuídas, a busca foi orientada em três dimensões de conhecimento:

- Contexto (C): compõe-se de diferentes interpretações a um domínio de conhecimento, constituindo-se de representações ou abordagens diferentes de um mesmo objeto de conhecimento.
- Profundidade(P): relativa a um contexto particular, refere-se a um refinamento na linguagem de percepção, ou seja, estratificação dos vários níveis epistemológicos de percepção do objeto de conhecimento.
- Lateralidade(L): referente aos conhecimentos intimamente relacionados ao domínio de aplicação, proveniente de uma visão particular de contexto e profundidade.

Pode-se pensar em uma visão interna de cada subdomínio referente a um domínio de conhecimento constituído por um conjunto de unidades pedagógicas, definidas em função de objetivos de ensino/aprendizagem específicos, associados a um *curriculum*. As unidades pedagógicas são relacionadas segundo uma ordem definida com base em critérios

pedagógicos e a cada uma corresponde um conjunto de problemas. A cada problema está associado um conhecimento de suporte à sua resolução, que pode incluir conceitos, exemplos, contra-exemplos, dicas etc.

A adoção de uma abordagem baseada em agentes deve-se ao fato de que tal abordagem se mostra adequada ao tratamento da complexidade envolvida no conhecimento e sua manipulação, e também devido às características do modelo do domínio.

4.3.1 Interações no MATHEMA

Inicialmente, pode existir um aprendiz interessado em trabalhar no ambiente MATHEMA, que foi incentivado pelo Motivador externo. A esta situação inicial segue-se uma interação entre o aprendiz e o agente de Interface (AI), sendo que o aprendiz informa ao AI os seus objetivos. O AI por sua vez, deve prover informações sobre o ambiente computacional e deve auxiliá-lo mediante análise de seus objetivos a escolher um supervisor na SATA. A partir deste ponto, inicia-se um processo de interação cooperativa e didática entre o aprendiz e o agente Supervisor, sendo que este passa a ser o responsável pelo aprendiz.

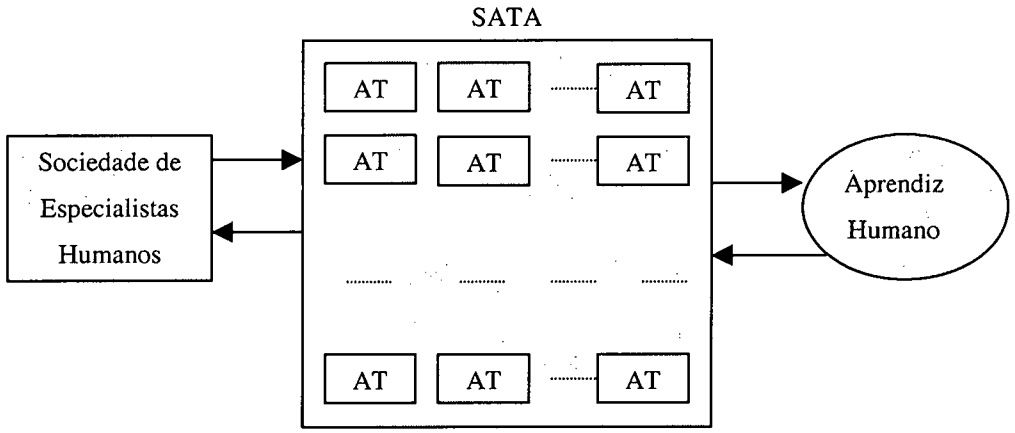

Figura 4-1 Interações no MATHEMA

Durante a interação, situações com diferentes níveis de complexidade podem ocorrer.

Constata-se que a sociedade de agentes não tem o conhecimento necessário para realizar determinada atividade, assim, notifica-se o aprendiz a respeito da impossibilidade ou dificuldade de atender sua solicitação e contata-se a Sociedade dos Especialistas Humanos para que sejam feitas as intervenções necessárias para a correção do problema. A SEH deve ser informada via agente de Manutenção, e tendo solucionado o problema, deve notificar o agente Supervisor.

No sistema há essencialmente três tipos de interações:

- Interações entre um aprendiz e a SATA através de um agente tutor.
- Interações entre os agentes tutores.
- Interações entre SATA e SEH.

As interações dinâmicas entre um aprendiz e um agente tutor ocorrem num contexto de resolução de problemas, ativando diferentes funções pedagógicas durante o processo adaptativo. O agente responsável pela interação com o aprendiz pode solicitar a cooperação de outros agentes quando verificar que não pode realizar determinadas tarefas. As interações entre as sociedades SATA e SEH ocorrem quando um agente tutor faz alguma solicitação a SEH ou quando a SEH constata que existe necessidade de intervenções. Esta constatação pode vir da observação do processo interativo entre aprendiz e SATA, a partir de uma determinada fonte, podendo esta ser algo como um mecanismo de registro.

4.4 Sociedade de agentes tutores artificiais

4.4.1 Modelo de agente tutor

Os agentes cooperam entre si através de linguagens e protocolos, a fim de oferecer ao aprendiz condições mais efetivas no suporte às atividades de aprendizagem. São agentes do tipo inteligentes (ou cognitivos) e são assumidos como benevolentes, pois concordam em cooperar com os demais agentes quando solicitados.

Cada agente é uma entidade especializada em um domínio específico possuindo como principais características autonomia, onde cada agente têm seus próprios objetivos; habilidade social, onde os agentes interagem entre si e com o mundo externo à SATA e ainda são orientados a objetivos, onde cada agente exibe um comportamento de acordo com suas capacidades.

A arquitetura de um agente tutor, segundo Costa [15], pode ser vista em dois níveis distintos de abstração: *macro* e *micro*.

A Figura 4-1 mostra a arquitetura geral de um agente tutor no nível macro.

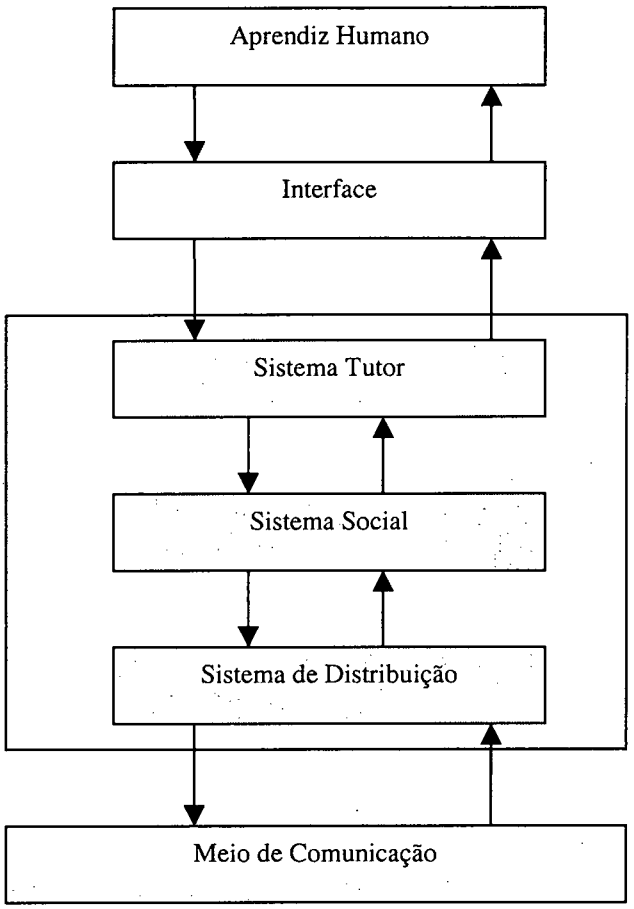

Figura 4-1 Agente Tutor no Nível Macro

Segundo o nível macro um agente é composto por três componentes principais:

- Sistema Tutor: interage diretamente com o aprendiz e armazena os conhecimentos que o agente possui para efetuar operações pedagógicas no domínio de aplicação.
- Sistema Social: possui bases de conhecimento e mecanismos de raciocínio necessários ao comportamento cooperativo entre os agentes tutores, refletindo o comportamento social do agente.
- Sistema de Distribuição: manipula a troca de mensagens entre agentes, através do meio de comunicação. Gerencia internamente a distribuição das mensagens no agente tutor.

Uma visão interna da arquitetura de um agente tutor é apresentada num nível micro. A Figura 4-2 ilustra uma visão interna dos componentes do sistema.

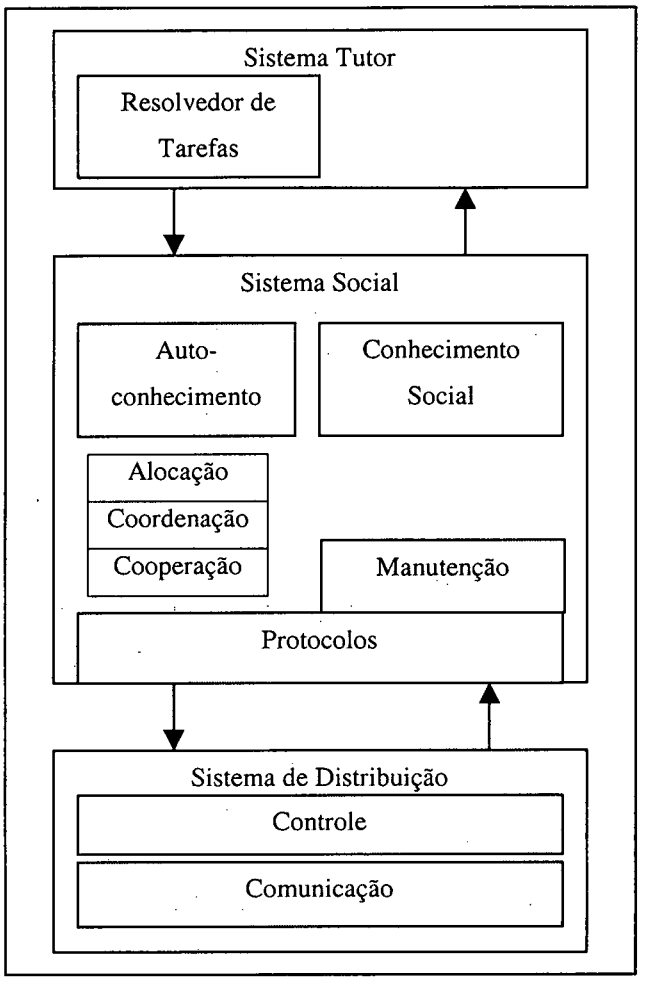

Figura 4-2 Agente Tutor no Nível Micro

O sistema tutor é formado por um módulo denominado Resolvedor de Tarefas, que é responsável por desempenhar um conjunto de tarefas pedagógicas e identificar aquelas que não são de sua competência.

O sistema social engloba:

- Autoconhecimento é a estrutura de conhecimento que o agente tem de si mesmo;
- Conhecimento social é o conhecimento sobre habilidades dos outros agentes tutores da sociedade.

Estes módulos de conhecimento são consultados sempre que é necessário identificar um agente que possa resolver uma determinada tarefa pedagógica.

- Alocação é o módulo que efetua a seleção dos agentes aptos a resolver uma determinada tarefa, para a cooperação.
- Coordenação é um mecanismo que interpreta a estrutura de tarefas decompostas.

- Cooperação é responsável por promover a execução de uma tarefa, gerenciamento dos diálogos, encaminhando-a para algum agente da sociedade (inclusive ele próprio).
- Manutenção é responsável pelo gerenciamento dos diálogos de manutenção, permitindo a realização de operações que podem incluir entrada, saída e atualização de agentes, com a finalidade de manter os conhecimentos sociais atualizados.
- Protocolos é o responsável pela criação e ativação de um diálogo com base nos protocolos disponíveis. Os protocolos disponíveis definem um comportamento interativo, enquanto que sua instância é uma entidade com um contexto particular responsável pela sua execução.

O sistema de distribuição é definido por dois módulos:

- Controle é responsável pela intermediação entre o Sistema Social e o Sistema de Distribuição.
- Comunicação realiza a distribuição e coleta de mensagens, fazendo a mediação entre o Sistema de Distribuição e o meio de comunicação. O modelo MATHEMA define detalhadamente a funcionalidade das atividades cooperativas.

O MATHEMA inspira-se numa abordagem construtivista, recebendo também influências da teoria de Vygotsky, a respeito dos aspectos sociais envolvidos no processo de interação, combinando, em seu modelo de ensino-aprendizagem, o qual é cooperativo, a aprendizagem pela ação e por instrução. A aprendizagem pela ação enfatiza características exploratórias em situações de resolução de problemas, conduzindo a uma forma de aprendizagem por descoberta a partir da ação, inspirada de um certo modo, na categoria micromundo. A aprendizagem por instrução tem um aspecto mais instrucional, proveniente dos STI's clássicos.

4.4.2 Arquitetura de um Sistema Tutor

Formalmente, um sistema tutor é definido como:

ST = {Med, Raciocinadores, Bases de Conhecimento}

onde:

- *Med* representa o módulo Mediador, um mecanismo pedagógico que deve reagir a cada ação vinda do aprendiz.
- *Raciocinadores* são os mecanismos provedores de funções pedagógicas.

- *Bases de Conhecimento* representa o módulo que mantém os conhecimentos de suporte à operacionalização dos raciocinadores (incluindo os conhecimentos sobre a estrutura pedagógica (*curriculum*), o domínio e o aprendiz).

O módulo Raciocinadores possui três submódulos: Especialista, Tutor e Modelagem do Aprendiz. O primeiro deles é um sistema especialista que realiza funções como resolução de problemas, tanto aqueles colocados ao aprendiz quanto aqueles propostos por este. Possui os submódulos: Resolvedor de Problemas, Explicador, Diagnóstico Cognitivo e Remediação. Tutor é o módulo que seleciona os recursos pedagógicos, escolhidos sobre o *curriculum*, e realiza atividades instrucionais. O módulo de Modelagem do Aprendiz constrói o modelo do aprendiz, contando com resultados do módulo de diagnóstico cognitivo. É de sua responsabilidade identificar os acertos, conhecimentos adquiridos, erros e mau entendimentos apresentados pelo aprendiz. O módulo Mediador decide sobre o tipo de intervenção a ser realizada para atender ao aprendiz, possuindo conhecimentos sobre os Raciocinadores e as Bases de Conhecimento. Este módulo tem também o papel de interação com o Sistema Social.

Cada agente possui um sistema tutor inteligente associado, responsável por interagir diretamente com o aprendiz. Neste sistema o domínio de conhecimento, o aprendiz e ainda outros componentes são modelados de forma distribuída, buscando atingir um processo educacional que gere resultados efetivos, através de uma solução eficaz para a questão da adaptabilidade do sistema ao estado cognitivo do aprendiz.

Com base na modelagem do conhecimento, a Sociedade de Agentes Tutores é definida. a cada subdomínio de um domínio D é associado um agente específico, obedecendo a seguinte relação:

$$D_{ij} \rightarrow AT_{ij} \qquad 4\text{-}1$$

A mesma idéia é considerada no tratamento do conhecimento lateral, ou seja, a cada visão de lateralidade é atribuído um agente:

$$dl_{ijk} \rightarrow AT_{ijk} \qquad 4\text{-}2$$

Sendo assim, a sociedade de agentes tutores artificiais (SATA) em relação a um domínio de conhecimento D é definida como:

$$SATA = AT \cup ATL \qquad 4\text{-}3$$

onde AT é o conjunto dos agentes tutores relativos a um domínio D, e ATL representa o conjunto dos agentes tutores associados a um domínio DL.

Um sistema tutorial de forma geral é constituído de quatro módulos básicos: Conhecimento do Domínio, Estratégias Didáticas, Modelo do Aluno e Interface. Nesta proposta, o Conhecimento do domínio pode ser analisado levando-se em consideração dois aspectos: a *profundidade* e o *contexto*. Fala-se em profundidade no sentido em que um domínio pode ser constituído de vários níveis, pois um determinado conhecimento pode ser ensinado em vários níveis que irão depender, por exemplo, do grau de conhecimento do usuário sobre o assunto em questão e sobre as técnicas necessárias ao tratamento dos problemas no domínio. O contexto diz respeito às várias interpretações que podem ser dadas quando se quer ensinar alguma coisa. Por exemplo, a dedução de alguma fórmula pode ser feita utilizando-se propriedades distintas, mas que levem ao mesmo objetivo final. Um outro exemplo é o ensino de Mecânica dentro da Física pode ser feito utilizando-se a formulação Newtoniana ou ainda o conceito de energia. Ambos os contextos levam ao objetivo comum que é o entendimento da Mecânica.

Didática diz respeito às técnicas em que será baseado o ensino e este módulo está intimamente ligado com o Modelo do Aluno. Esta área ainda apresenta diversos problemas em aberto. A arquitetura adotada, ao dividir o domínio de acordo com as dimensões definidas acima --profundidade e contexto -- e associar a cada ponto neste espaço um tutor inteligente especializado permite a definição de critérios para orientar a definição de um modelo do aluno.

4.5 Conclusão

O objetivo deste capítulo foi apresentar de forma resumida o modelo MATHEMA [15], para que o leitor possa compreender e situar este trabalho num âmbito global. Acredita-se que o conteúdo apresentado foi suficiente para alcançar este objetivo, todavia caso haja necessidade de aprofundar no assunto pode se obter estas informações na tese de Evandro de Barros Costa [15].

No Apêndice C está a modelagem do conhecimento relativa ao domínio de Estrutura da Informação, retirada da dissertação de Terezinha de Fátima Faria [23].

5 MathTutor

5.1 Introdução

Na concepção e desenvolvimento de softwares educacionais, além de possuir uma forma atraente precisa ser interativo e adaptar-se ao aprendiz.

O MathTutor foi desenvolvido visando os objetivos do parágrafo anterior. É um Sistema Tutor Inteligente (STI) baseado numa arquitetura multiagente, onde cada agente possui o conhecimento de parte do domínio e que pode ser acessado via Internet, experimentalmente, com opções de navegação monitorada ou não pelo próprio sistema.

O MathTutor pode ser visualizado em navegadores comuns, como Netscape Navigator ou Microsoft Internet Explorer, pois foi desenvolvido em Java.

5.2 Quadro Comparativo

De acordo com as propostas de trabalhos futuros na dissertação de Terezinha de Fátima Faria [22], podemos fazer um paralelo do estado inicial do projeto e o estado atual verificando as etapas superadas como mostra o quadro a seguir.

Antes	Agora
Servidor simulado	Servidor real operando
Sistema de arquivos para armazenamento de dados	Banco de Dados
Interface simples	Interface amigável
Esquema do agente sem efetiva integração com o sistema	Comunicação entre agentes integrados ao sistema e desenvolvimento do agente de interface
Disponibilidade local	Disponibilidade via Internet

Tabela 5-1 Quadro Comparativo

Os resultados obtidos com cada uma destas alterações são detalhados a seguir.

5.2.1 Servidor de Páginas *Web*

O servidor de páginas *web* que antes era um simulador passa a ter um servidor real operando. Com o servidor de páginas Apache [1] Tomcat [31], podemos acessar o curso através da porta 8080 da máquina em que o sistema está armazenado. O software Tomcat é uma implementação de Servlet, escrito em Java.

O funcionamento básico do servidor de páginas *web* é: o servidor Apache encaminha solicitações URLs correspondentes aos Servlets e estas enviam ao Tomcat para o processamento.

O funcionamento dos Servlets será abordado ainda neste capítulo, em maiores detalhes.

5.2.2 Sistema de Arquivos

Todas as informações sobre o sistema, tais como as informações que são armazenadas do aprendiz e do professor, por exemplo: páginas acessadas, exercícios efetuados, alunos matriculados, dados pessoais, entre outros, eram guardados em arquivos específicos. Isto fazia com o sistema se tornasse bastante extenso.

Com a inserção do banco de dados, o MathTutor teve seu número de arquivos reduzido e o acesso aos dados facilitado.

O sistema gerenciador do banco de dados é o PostgreSQL [46], que suporta sintaxe SQL e é de livre distribuição.

5.2.3 Interface

Inicialmente a interface do sistema foi deixada em segundo plano, pois a preocupação era com a funcionalidade e operação do sistema.

Já na segunda fase do projeto, uma das metas estabelecidas era tornar a interface mais amigável. Através de conhecimentos básicos de ergonomia construiu-se o formato atual da página.

5.2.4 Agentes

Uma arquitetura da operação dos agentes já estava previamente definida e alguns testes, de comunicação entre os agentes, haviam sido realizados com o simulador.

Agora se tem a integração efetiva dos agentes com o sistema e a troca de mensagens entre eles estabelecida. Os agentes trocam mensagens, que são recebidas e tratadas pelo agente de interface ou pelos outros agentes (teórico, prático – ver modelagem do conhecimento do domínio no Apêndice C).

5.2.5 Disponibilidade

Com o servidor operando o MathTutor passa a estar disponível pela Internet através da rede do DAS (Departamento de Automação e Sistemas) - http://baker.das.ufsc.br:8080/tutor - e posteriormente, quando novas funcionalidades forem adicionadas ao sistema e o sistema tenha um formato mais direcionado ao ensino à distância ou servir como um programa de autoria, facilitando o ensino de qualquer outra disciplina pela rede.

Além dos aspectos citados anteriormente, outras modificações foram feitas, podendo destacar ainda:

- Padronização das páginas a serem apresentadas.
- Padronização na inserção de conteúdo.
- Separação física dos arquivos que formam a base de conhecimento, dos arquivos que formam a interface.
- Alterações da base de regras que compõem o MathTutor na medida que o STI foi evoluindo e solicitando novas funções e adaptações.

5.3 Ferramentas

Conforme descrito anteriormente, o STI proposto utiliza técnicas de Inteligência Artificial Distribuída seguindo a abordagem de sistemas Multiagente cognitivos. A principal razão para utilizar a tecnologia de agentes em um STI está nas propriedades que

permitem ganho de qualidade sob o ponto de vista pedagógico, por exemplo, a habilidade social e a flexibilidade. Utilizar agentes em um programa educacional permite intensificar os aspectos pedagógicos desejáveis no ambiente [26].

A primeira decisão de projeto para a implementação desta estrutura multiagente foi a adoção da linguagem de programação Java. Esta decisão visa a portabilidade do sistema e sua independência em relação a plataformas de hardware e software, além da facilidade de comunicação com aplicações convencionais (como banco de dados e sistemas de rede). Uma vez definida a linguagem de programação, buscaram-se ferramentas de domínio público com as características desejadas. As ferramentas escolhidas foram: JESS (Java Expert System Shell) [25] para o suporte de Inteligência Artificial, JATLite (Java Agent Template Lite) [32] para a comunicação dos agentes, Java *Servlets* [33] e o protocolo HTML para o suporte multimídia.

O comportamento dos agentes do sistema é controlado por SE's. A ferramenta utilizada para desenvolver estes sistemas foi o JESS, um arcabouço para SE's escrito em Java que, além de permitir o desenvolvimento de sistemas baseados em regras, apresenta um alto grau de integração com a linguagem Java.

O JATLite utilizado para a comunicação dos agentes, suporta a linguagem KQML, que permite a troca de mensagens de acordo com a teoria dos atos de fala [50]. Os conteúdos destas mensagens são fatos que serão inseridos na máquina de inferência do JESS, permitindo que o sistema tome decisões e mude seu comportamento enquanto o usuário está interagindo com os *Servlets*. Todas as páginas de interface retornam para o usuário uma resposta dos *Servlets*, permitindo que o sistema saiba exatamente que página o usuário está acessando e que tipo de informação ele está obtendo. Estes assuntos serão abordados com mais detalhes nas próximas subseções.

5.3.1 Suporte de Inteligência Artificial

A capacidade cognitiva dos agentes da SATA é implementada através de um Sistema Especialista [63] formado por uma base de regras, uma memória de trabalho e um motor de inferência. As regras e os dados da memória de trabalho, aos quais estas são aplicadas, formam uma base de conhecimento.

Os sistemas especialistas do MathTutor são implementados utilizando JESS, que é utilizado em muitas aplicações, porém em algumas delas ele não possui um comportamento eficiente, como quando o utilizamos com *applets*. O tamanho do JESS torna o sistema muito pesado para o uso de *applets* [25]. Quando a idéia é utilizar aplicações com JESS via navegador, devemos considerar o uso do JESS do lado do servidor, como no caso dos *Servlets*. Os *Servlets* permitem que apenas o resultado seja enviado ao aprendiz, dispensando a necessidade do usuário ter que carregar grande parte do sistema para a sua máquina, o que torna a interação com o sistema bastante lenta e entediante do ponto de vista do aprendiz.

Aplicações JESS podem ser escritas e controladas por um código Java, através do uso da biblioteca do JESS, que é o utilizado no MathTutor. Existem duas maneiras de utilizarmos Java com o código JESS: Java pode estender o JESS ou fazer uso da sua biblioteca [25].

Através de algumas suposições podemos exemplificar o funcionamento da base de regras JESS.

Caso o aprendiz queira buscar por um determinado assunto no sistema, ele digita uma palavra ou uma frase. A partir daí, uma procura será feita através de palavras-chave extraídas da frase, o JESS verifica qual ou quais das palavras digitadas estão contidas na base de conhecimento do sistema e mostra ao aprendiz a página ou as páginas relacionadas ao conteúdo solicitado. Na Figura 5-1 podemos visualizar o resultado da busca pela palavra "iteração".

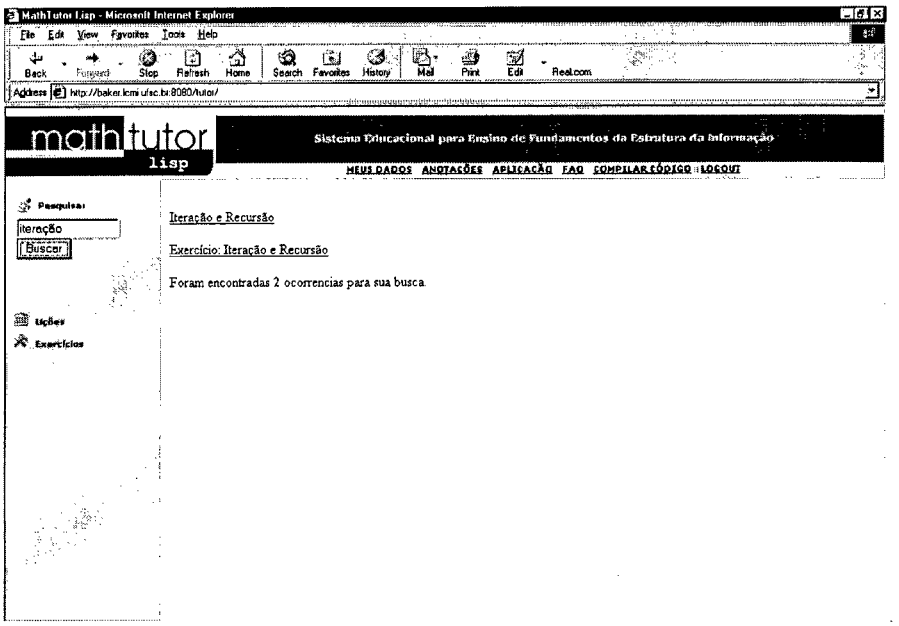

Figura 5-1 Interface com Resultado de uma Busca

Os exercícios apresentados ao aprendiz devem ser feitos em Lisp [13]. Para implementar a interação JESS-Lisp, foi preciso definir um "simulador", que faz com que o Lisp leia os dados de entrada de um arquivo onde são gravadas as funções criadas pelo usuário, realizando a interpretação. Os resultados são então armazenados em arquivos específicos, sendo um destinado aos dados referentes a funções sintaticamente corretas e, outro destinado às mensagens geradas devido a erros. Estes arquivos são então lidos por métodos responsáveis em retornar ao aprendiz, através da interface, os resultados da interpretação dos algoritmos por ele apresentados [22]. Atualmente, a equipe de desenvolvimento está trabalhando para alterar o modo de armazenamento dos dados gerados pelo interpretador Lisp, de um sistema de arquivos para um banco de dados, facilitando a visualização, organização e monitoração dos caminhos seguidos para cada aprendiz e sua evolução no curso.

5.3.2 Suporte de comunicação de agentes

O suporte de comunicação entre os agentes do MathTutor é implementado na linguagem Java com auxílio do JATLite. O JATLite foi desenvolvido na Universidade de Stanford e fornece um modelo funcional para a construção de sistemas multiagentes usando o protocolo de rede TCP/IP e o protocolo de comunicação entre agentes KQML. O JATLite pode ser definido como um conjunto de classes escrito em Java que provê uma arquitetura básica para a construção de agentes que se comunicam através da rede Internet.

O JATLite é formado por cinco camadas, sendo elas:

- Camada abstrata
- Camada de base
- Camada KQML
- Camada do roteador
- Camada de protocolo

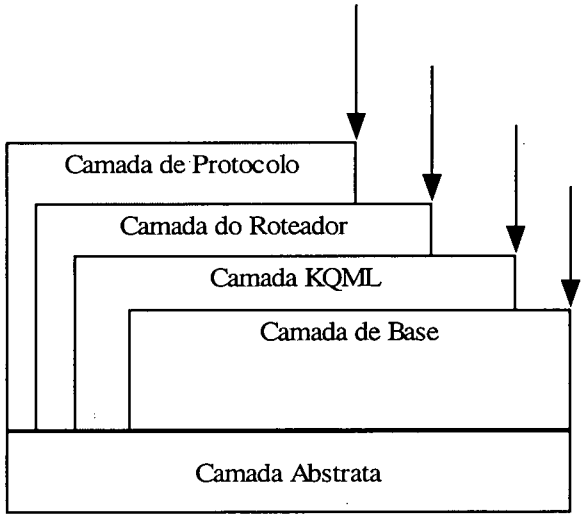

Figura 5-1 Camadas do JATLite

Cada uma dessas camadas possui um conjunto de classes que podem ser reutilizadas pelos usuários desenvolvedores. Os agentes podem ser desenvolvidos como aplicações JAVA que rodam independentemente da presença do navegador *web* ou do servidor *web*.

A camada do roteador garante a presença de um agente especial, conhecido como Roteador, que atua como servidor armazenando os nomes de todos os agentes que fazem parte do sistema. Conhecendo-se o endereço dos agentes, a entrega das mensagens aos

destinatários é facilitada. A transmissão das mensagens entre os agentes pode se dar através de um mecanismo de *pooling* onde o agente emissor verifica se o receptor está conectado e envia a mensagem. As conexões entre os agentes são feitas de uma maneira persistente, ou seja, a conexão fica ativa até que o agente resolva fechá-la ou um tempo máximo de ociosidade seja atingido (*timeout*). A segurança pode ser feita através da checagem do nome e senha dos agentes que querem se conectar ao sistema. Geralmente, quem realiza esse serviço é o agente Roteador já que é ele quem tem conhecimento de todos os agentes conectados.

A camada do roteador se caracteriza pela presença do agente Roteador que presta vários serviços aos agentes conectados a ele. Tais agentes enviam as mensagens para o Agente Roteador, e este as remete para o endereço correspondente ao nome do agente receptor. Se por algum motivo o agente receptor não puder receber a mensagem de outro agente, esta será armazenada em arquivos até o momento em que esse agente entre em operação normal.

Uma das maneiras de se desenvolver um sistema Multiagente, através da JATLite, é através da camada do roteador. Essa camada traz facilidades no sentido de abstrair detalhes de programação de baixo nível podendo-se reutilizar toda a infra-estrutura das camadas inferiores. O MathTutor possui seus agentes desenvolvidos nesta camada pois utiliza os recursos das camadas inferiores tal como foram desenvolvidas como por exemplo, adotando a linguagem KQML.

No Apêndice D é mostrado o código de um dos agentes que formam o MathTutor exemplificando o uso do JATLite.

5.3.3 Suporte multimídia

O suporte multimídia do MathTutor é implementado utilizando *Servlets*. Os Java *Servlets* possuem um modelo de programação similar aos *scripts* de CGI[2] , na medida em que eles recebem uma solicitação de HTTP de um navegador *web* como entrada e espera-se que localizem e/ou construam o conteúdo apropriado para a resposta do servidor. Todos os *Servlets* associados a um servidor *web* rodam dentro de um único processo. Este

[2] CGI Common Gateway Interface

processo roda uma Máquina Virtual Java (Java Virtual Machine - JVM), que é o programa específico da plataforma para rodar programas Java compilados. Ao invés de criar um processo para cada solicitação, a JVM cria um encadeamento Java para tratar de cada solicitação de *Servlet*.

Já que a JVM persiste além de uma única solicitação, os *Servlets* também podem evitar muitas operações demoradas, como conexão a um banco de dados, ao compartilhá-los entre as outras solicitações. Pelo fato de serem escritos em Java, os *Servlets* aproveitam todos os benefícios da plataforma básica do Java de um modelo de programação orientado a objetos, gerenciamento automático de memória e portabilidade. Basicamente os *Servlets* fornecem uma metodologia baseada em Java para mapear solicitações de HTTP em respostas HTTP. A geração do conteúdo da *web* dinâmica usando *Servlets* é realizada através de código Java que fornece o código HTML representando aquele conteúdo [8].

No MathTutor a interação com o *Servlet* acontece sempre que ocorra uma troca de informação entre o sistema e o aprendiz. Podemos exemplificar da seguinte forma: quando um exercício é proposto e o aluno escreve numa área destinada ao texto, sua solução é enviada para um arquivo que utilizará o simulador do Lisp. O resultado desta simulação será apresentado para o aprendiz por um outro *Servlet* que usa Java para ler o conteúdo dos arquivos textos gerados pelo simulador. O aprendiz irá visualizar este resultado em uma página HTML.

Para a interface utilizamos o protocolo HTTP, que facilitará no futuro a utilização do sistema para ensino à distância. A funcionalidade de interface do sistema é implementada através de páginas *web*, utilizadas para desenvolver a interação entre o aprendiz e o sistema tutor. A interface provoca disparos das regras JESS e estas regras podem realizar funções como manipular os dados que compõem o modelo do aprendiz e o tratamento das habilidades de diagnóstico dos agentes, realizando tratamento de erros etc. A interação navegador-JESS foram solucionadas através do uso de *Servlets*, que tornou-se uma excelente solução devido à ausência de restrições de segurança tal como impostas pelos *applets*, pelo fato de que uma aplicação local (*Servlet*) é que define as páginas HTML, dentro do próprio código de classes Java que implementam o tutor. Além do fato de fazermos uso freqüente de acesso a um sistema de arquivos, o que não é algo trivial em *applets*.

Os *servlets* mapeiam solicitações de HTTP em respostas HTTP. A geração de conteúdo da *web* dinâmico usando-se *servlets* realizada através de código Java que fornece a HTML

representando aquele conteúdo. No caso dos dados de HTML, uma abordagem é fazer com que o código de Java construa cadeias de texto e depois imprima estas cadeias no fluxo de saída associado com a resposta de HTTP.

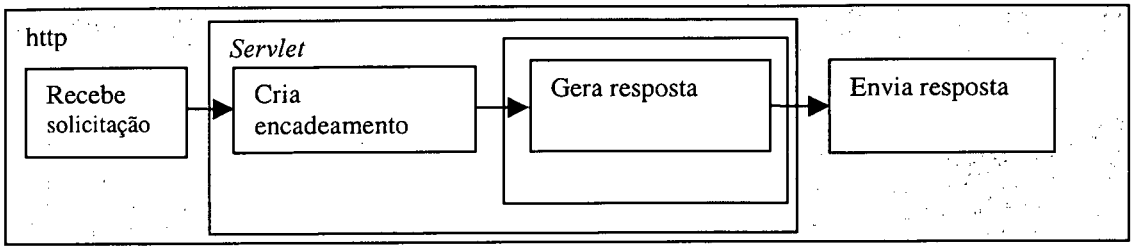

Figura 5-1 Processo de Servidor para executar Servlets

5.4 Arquitetura do MathTutor

A concepção de um processo ensino-aprendizagem é centralizada no especialista, conforme mostra a Figura 5-1, podendo os agentes inteligentes assumirem algumas das tarefas destinadas a eles. Os agentes estão organizados em sociedade onde existe um agente de interface responsável pela comunicação entre o aluno e o sistema. Através da interação entre os agentes, os alunos podem fazer as lições, resolver exercícios e navegar livremente pelo conteúdo.

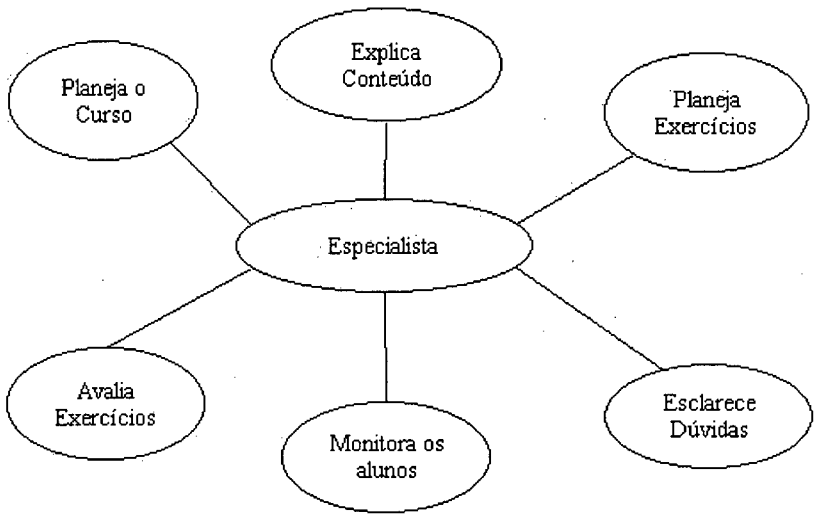

Figura 5-1 Funções do Especialista

O sistema apresenta o conhecimento e interage com o estudante através de um navegador da Internet, seguindo as opções pedagógicas que o sistema tutor decide. O conhecimento é modelado utilizando regras.

O tutor adquire conhecimento sobre o aprendiz através da sua interação com o navegador.

O sistema possui quatro módulos: o módulo do estudante, o módulo do especialista, o módulo pedagógico e o módulo da interface.

O módulo do especialista possui as informações a respeito do conhecimento do conteúdo a ensinar; a teoria e conceitos essenciais para que o estudante possa resolver um problema sem a ajuda do sistema.

O módulo do estudante armazena as informações sobre a compreensão do aluno sobre o domínio de conhecimento. Obtemos esta informação construindo um modelo de como o estudante avança na aprendizagem do curso utilizando ferramentas de diagnóstico contidas no modelo pedagógico.

O módulo pedagógico contém as regras para a tomada de decisão que permitem determinar o quanto o aluno está aprendendo.

O módulo da interface apresenta ao usuário o ambiente de aprendizagem desenvolvido pelo especialista.

A Figura 5-2 mostra a arquitetura do MathTutor:

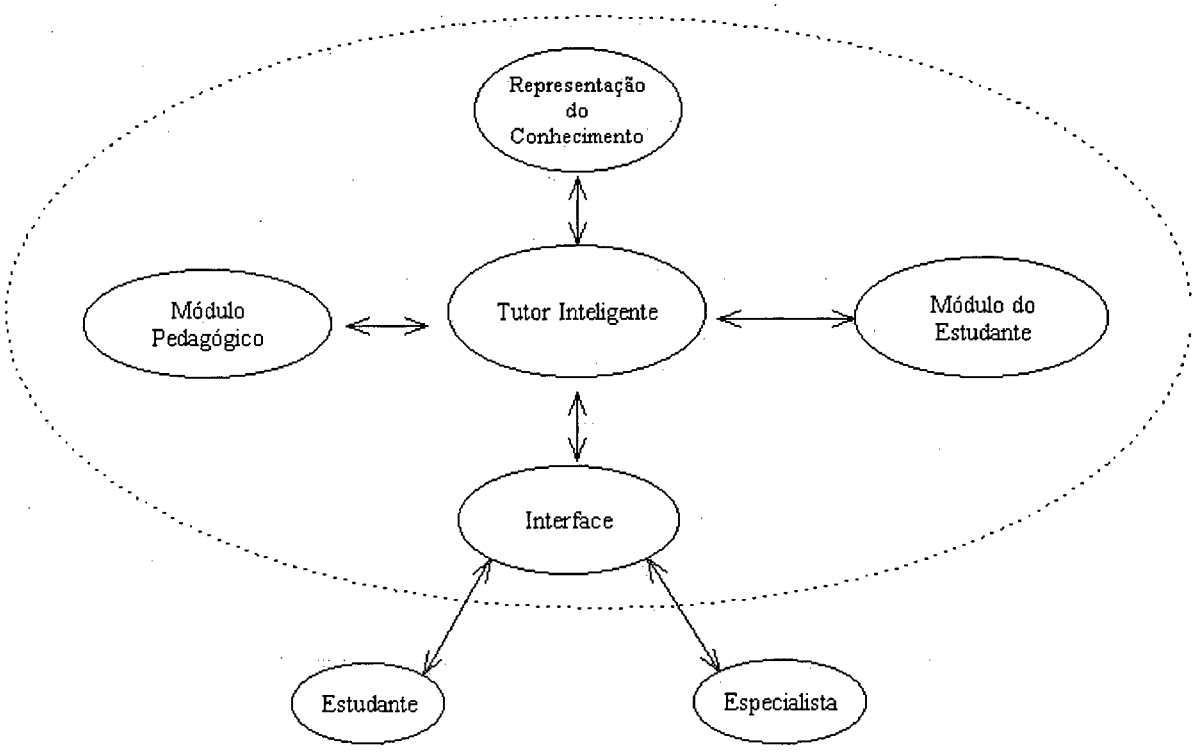

Figura 5-2 Arquitetura do MathTutor

Módulo do Estudante

Para podermos avaliar o que o aluno aprendeu é preciso interpretar seu avanço no curso. A única forma que o tutor interage com o aluno é por meio da interface, sendo necessário estabelecer algum critério de avaliação que observa entre outros aspectos:

- Se o estudante costuma rever o conteúdo
- Se faz os exercícios
- Visualiza os exemplos

Módulo do Especialista

O trabalho do especialista é elaborar as aulas, inserir o conteúdo referente a estas, construir a base de conhecimento para o sistema especialista e alimentar o MathTutor

através da busca na base de dados onde se armazena a informação de todo o trabalho realizado pelo estudante no curso. O especialista conta com um editor de conteúdo, que permite inserir textos, exercícios e, neste caso, o interpretador CLisp. Além disso, o especialista pode acessar uma área restrita e verificar os alunos que estão matriculados nas aulas e monitorar o desempenho deles.

Módulo Pedagógico

O módulo pedagógico toma as decisões sobre o que fazer a cada momento no tutor. A principal função do módulo pedagógico é aprofundar um tópico caso o sistema perceba que o estudante esteja respondendo os exercícios com relativa facilidade, ou caso ele tenha dificuldades deverá gerar um curso mais básico. Esta estrutura permite o sequenciamento curricular através das lições do tutor que determina o estilo de aprendizagem.

Alguns perfis são traçados de acordo com os perfis existentes de alunos. Há os que gostam de fazer só exercícios, outros que só estudam os exemplos e outros que seguem o curso tradicional, teoria, exemplo e exercícios. E a partir destes perfis os agentes decidem em qual o estudante se adapta naquele momento e sugerindo a próxima página.

Módulo da Interface

O Módulo da interface interage com o navegador da Internet. É formado por uma página com informação gerada ou fornecida pelos outros módulos. Está dividida em:

1. Busca
2. Navegação
3. Conteúdo
4. Índice das lições

A navegação tem dois botões que permitem: seguir para o próximo tópico da lição ou mudar de texto para exercício ou vice-versa. Além destes dois botões existem outros *links* que permitem navegar pelo MathTutor, conforme Figura 5-3.No índice está a estrutura convencional de um livro em que o estudante pode também navegar. Todas as ações do aprendiz são enviadas para o módulo do estudante permitindo a atualização das variáveis sobre o estudante e poder inferir melhor qual o percurso que o mesmo deve fazer pelo STI.

Estas ações são detalhadas na Seção 5.5.

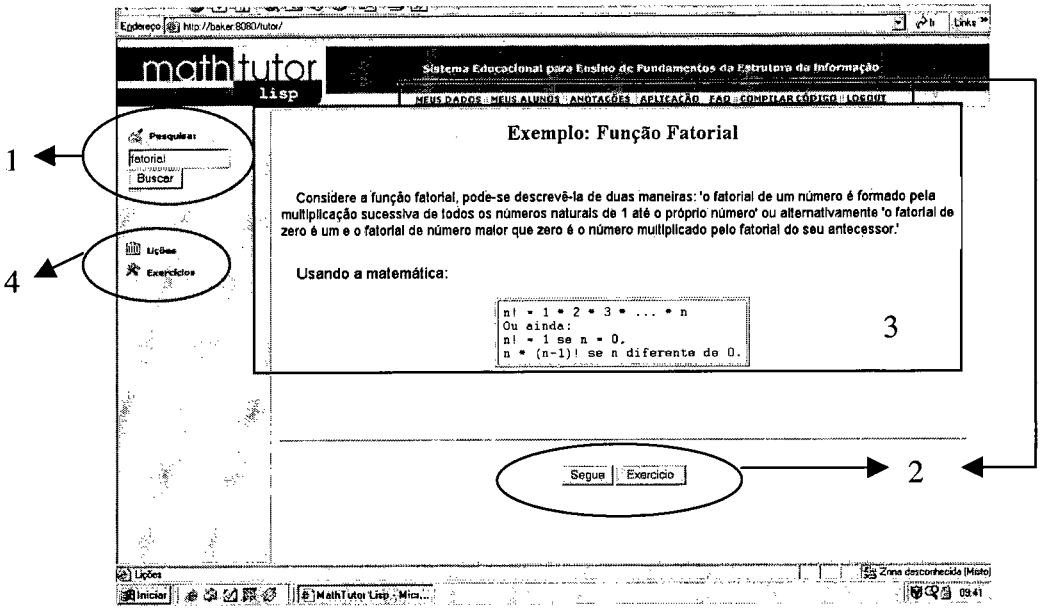

Figura 5-3 Interface

5.5 Interação com o Usuário

A interface com o usuário final é fundamental para o sucesso de um projeto computacional. Além de apresentar uma interface ergonomicamente bem projetada, deve-se levar em conta o grau de familiarização do usuário com o domínio de trabalho do sistema e com os computadores em geral. O aspecto crítico nesta etapa de apresentação do conteúdo é facilitar a compreensão, com explicações claras e diretas. Como já foi dito no início deste capítulo, existe sempre uma preocupação em desenvolver uma interface amigável e intuitiva ao usuário, acreditamos estarmos muito próximo do proposto.

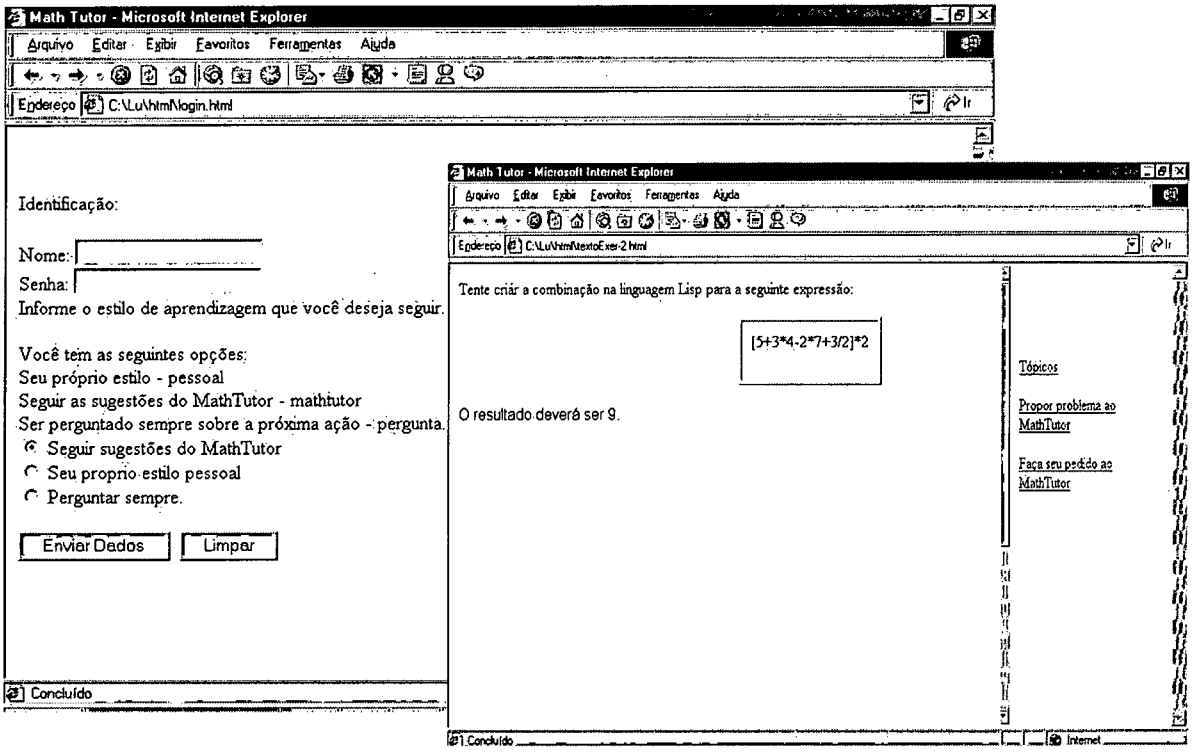

Figura 5-1 Interface Antiga

Além da interface, a maneira com que o conteúdo é apresentado também tem grande importância para o sucesso do sistema. Pensando nisto, e na facilidade que uma padronização na geração e na apresentação do conteúdo proporciona, foi desenvolvido um editor de conteúdo.

Todo o conteúdo a ser apresentado ao aprendiz está armazenado em um único arquivo.

O especialista insere o conteúdo do curso e a SATA organiza a ordem de apresentação para cada aprendiz.

As funções básicas do editor podem ser vistas no quadro a seguir:

```
-----> Editor MathTutor <------

* Comandos:

;;;;   -> Definir título do documento:   ">TITULO:" - seguido pelo titulo.
;;;;   -> Centralizar texto:             ">CENTRO:" - seguido pelo texto a ser centralizado.
;;;;   -> Novo parágrafo:                ">P:"      - seguido pelo texto de todo o  parágrafo
;;;;   -> Escrever texto em um quadro
;;;;     no centro da tela:              ">QUADRO<" - com o texto escrito nas próximas linhas
;;;;                                     até que uma instrução >FIM_QUADRO< seja encontrada
;;;;   -> Marcar o fim de um quadro:     ">FIM_QUADRO<"
;;;;   -> Inserir espaço para o aluno
;;;;     resolver um exercício proposto
;;;;     e enviar para interpretação:    ">COMPILAR_CODIGO<"
```

Quadro 5-1 Editor de texto do MathTutor

Um exemplo do uso do editor está apresentado a seguir:

```
EXEMPLO1
      >TITULO: Exemplo: Função Fatorial
>P: Considere a função fatorial, pode-se descrevê-la de duas maneiras: 'o fatorial de um número.....
>P: Usando a matemática:....

>QUADRO<
 n! = 1 * 2 * 3 * ... * n
 Ou ainda:
 n! = 1 se n = 0,
 n * (n-1)! se n diferente de 0.
>FIM_QUADRO<

;; Fim do Exemplo1
--->FimDoTexto<---
```

Quadro 5-2 Texto escrito segundo o editor

A apresentação final deste conteúdo pode ser vista a seguir:

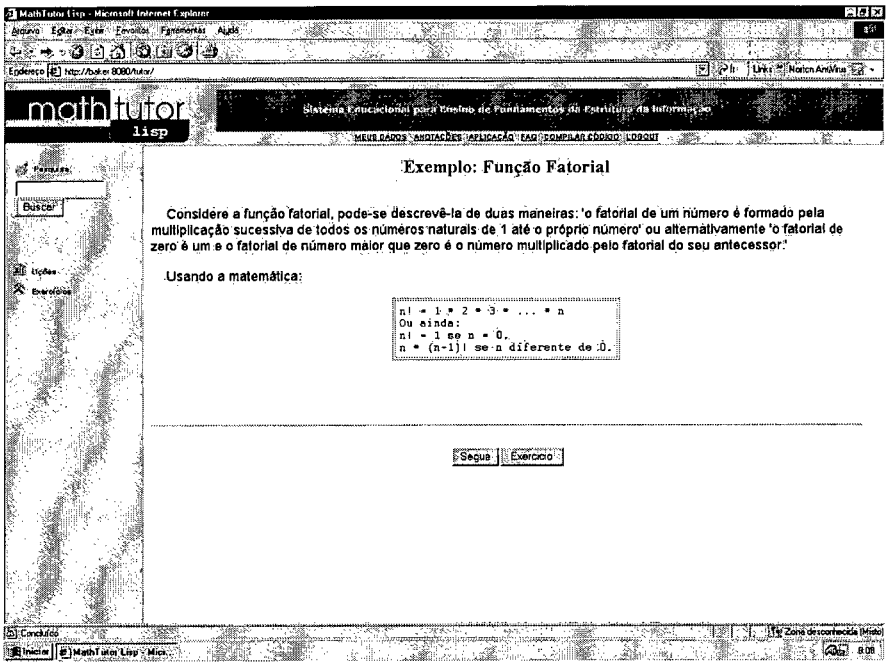

Figura 5-2 Página Resultante da Inserção de Conteúdo

A padronização da estrutura das páginas facilitará o uso do sistema para o ensino de outras disciplinas que não estejam obrigatoriamente relacionadas com o ensino de estrutura da informação, mas que possuem o mesmo público alvo, alunos de graduação ou pós-graduação.

O curso é composto por páginas de assuntos teóricos, exemplos e exercícios. A seguir serão apresentadas as principais páginas acessadas pelo aprendiz, o que irá familiarizar o leitor com o STI em questão.

Inicialmente o aluno se cadastra no sistema, criando uma senha pessoal de acesso. Após o cadastro ter sido efetuado o aluno estará apto a utilizar o MathTutor.

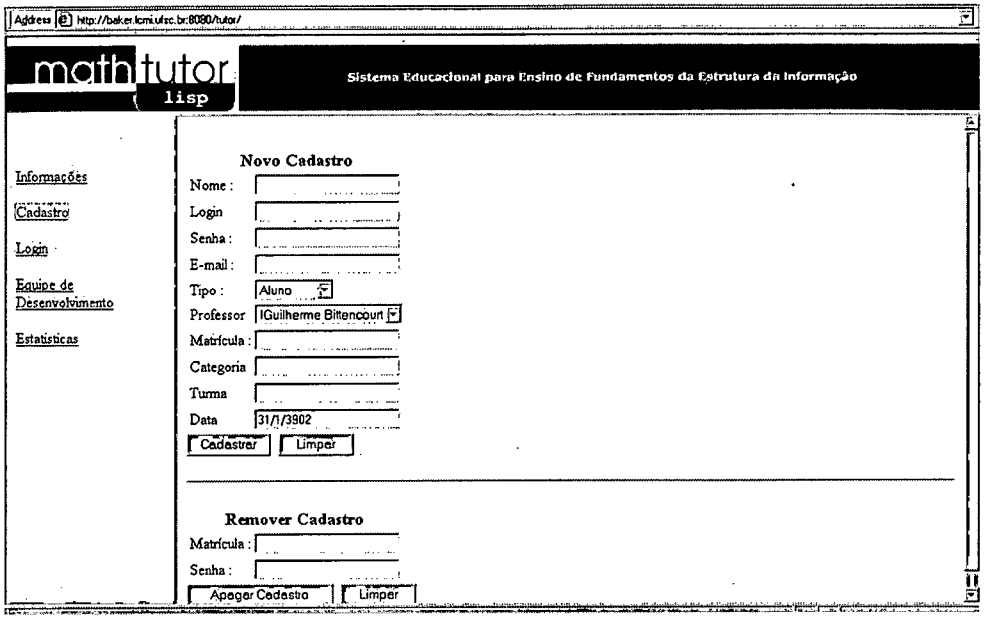

Figura 5-3 Página de Cadastramento no MathTutor

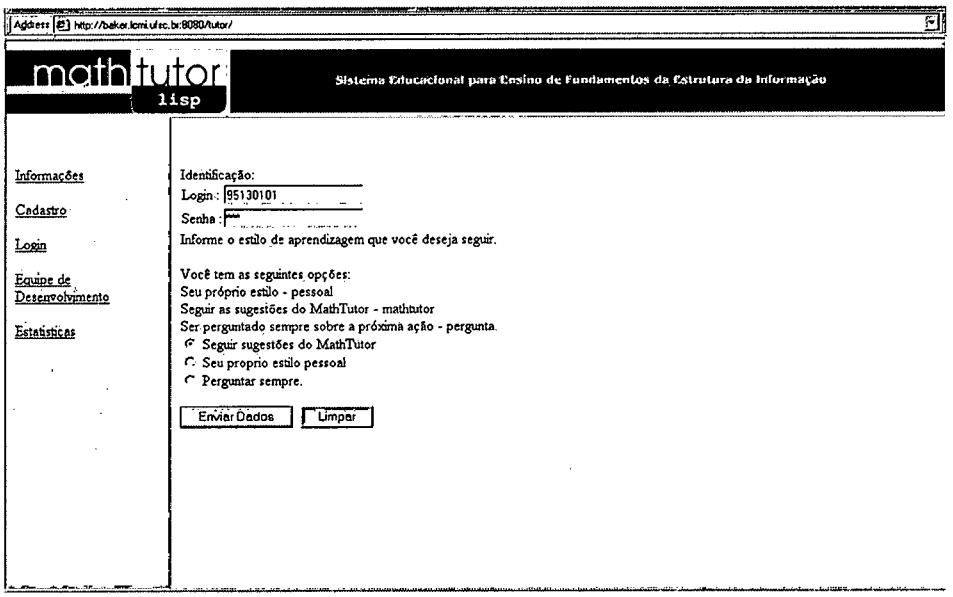

Figura 5-4 Login do MathTutor

Após efetuar seu acesso no sistema, o aluno poderá: visualizar e alterar seus dados cadastrais, fazer anotações diversas, que ficarão armazenadas no banco de dados do MathTutor, solicitar a interpretação de um código que não esteja relacionado a um exercício proposto e ainda buscar no sistema um assunto de seu interesse.

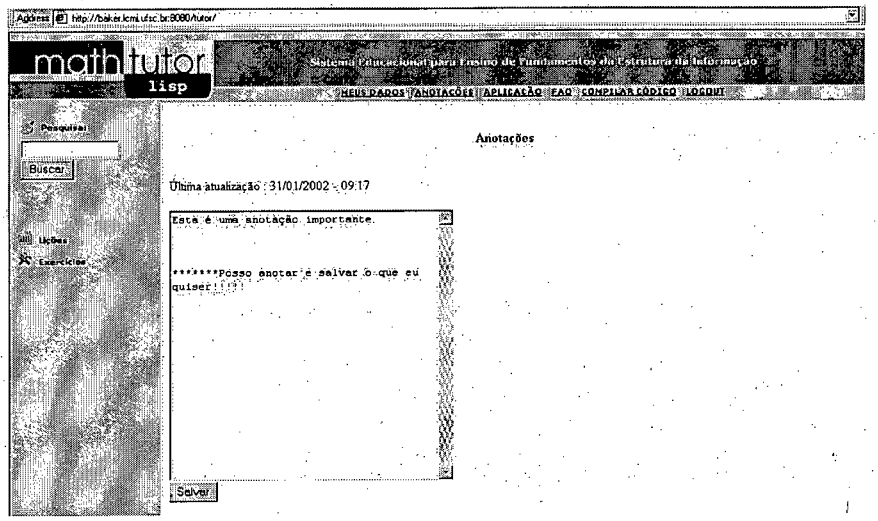

Figura 5-5 Página para Efetuar Anotações

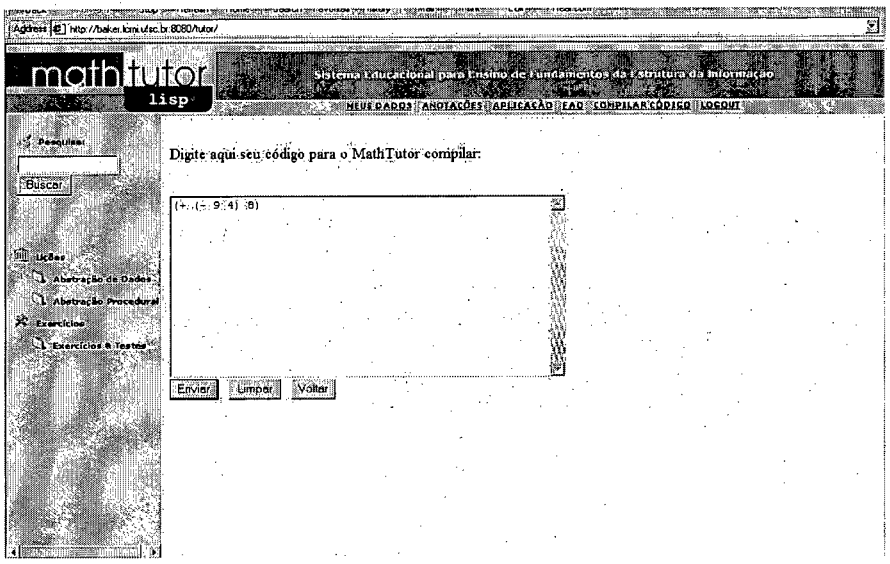

Figura 5-6 Interpretação de um código

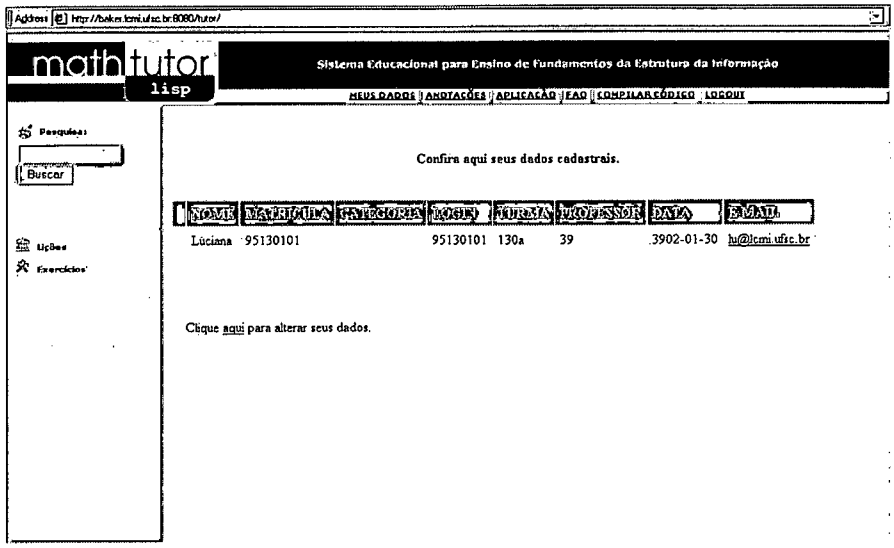

Figura 5-7 Visualização dos dados pessoais cadastrados

O professor pode verificar através dos dados, apresentados em uma área específica, o desempenho dos alunos e da turma como um todo.

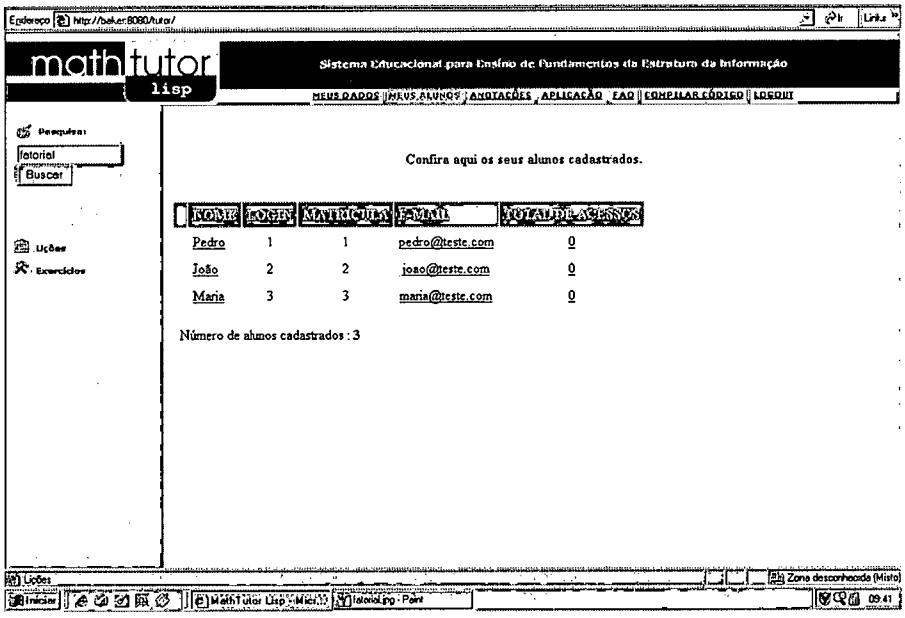

Figura 5-8 Área Exclusiva para Categoria Professor

Resumidamente podemos verificar as etapas de acesso ao sistema pelo aprendiz:

Um aluno faz seu cadastro e recebe uma mensagem na sua caixa de correio eletrônico confirmando seu cadastramento e passando a acessar o material do curso. A liberação ao conteúdo é feita através dos seguintes passos:

1. O pedido é enviado pelo navegador ao servidor.
2. O servidor reconhece o pedido.
3. Verifica a autenticação do usuário.
4. Confere se o usuário tem permissão de acesso para visualizar a página.
5. Carrega o conteúdo da página.

Caso o aprendiz já tenha acessado o MathTutor anteriormente, o sistema identifica qual a página foi acessada pela última vez e a mostra.

A modelagem destas e de outras operações do sistema podem ser visualizadas através dos diagramas no Apêndice B deste trabalho.

5.6 Comunicação

Durante todo o desenvolvimento do MathTutor a principal preocupação foi a comunicação, ou seja, como seria a troca de informação entre as ferramentas que compõem o sistema, entre o aprendiz e o STI, entre o especialista e o STI, entre os agentes, etc.

Sendo assim verificamos que a comunicação do sistema não se limita aos agentes, ela também ocorre entre os outros componentes e ferramentas do mesmo. Por exemplo, o banco de dados é acessado a todo instante, seja por solicitação do usuário ou pela base de conhecimento do MathTutor que necessita de alguma informação para executar sua função. Com base nestas informações podemos esquematizar o processo de comunicação do MathTutor da seguinte maneira.

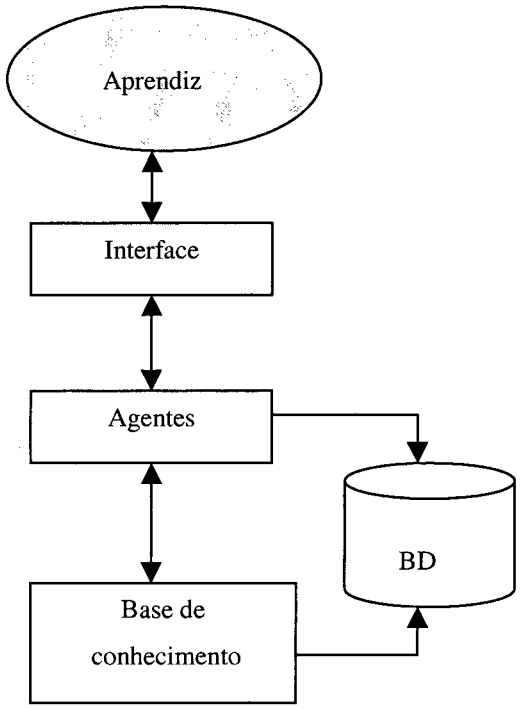

Figura 5-1 Esquema da comunicação entre os principais componentes do sistema

Os componentes vitais para o funcionamento e caracterização do MathTutor como um STI são: a base de conhecimento e os agentes. O MathTutor é formado por quatro agentes, onde dois deles são responsáveis pelo conteúdo teórico e os outros dois pelo prático, segundo a modelagem do domínio (ver Apêndice C). Atualmente temos um módulo em funcionamento resultando em apenas um agente de cada tipo.

Os agentes se comunicam com o ambiente externo através do agente interface como representado na Figura 5-2:

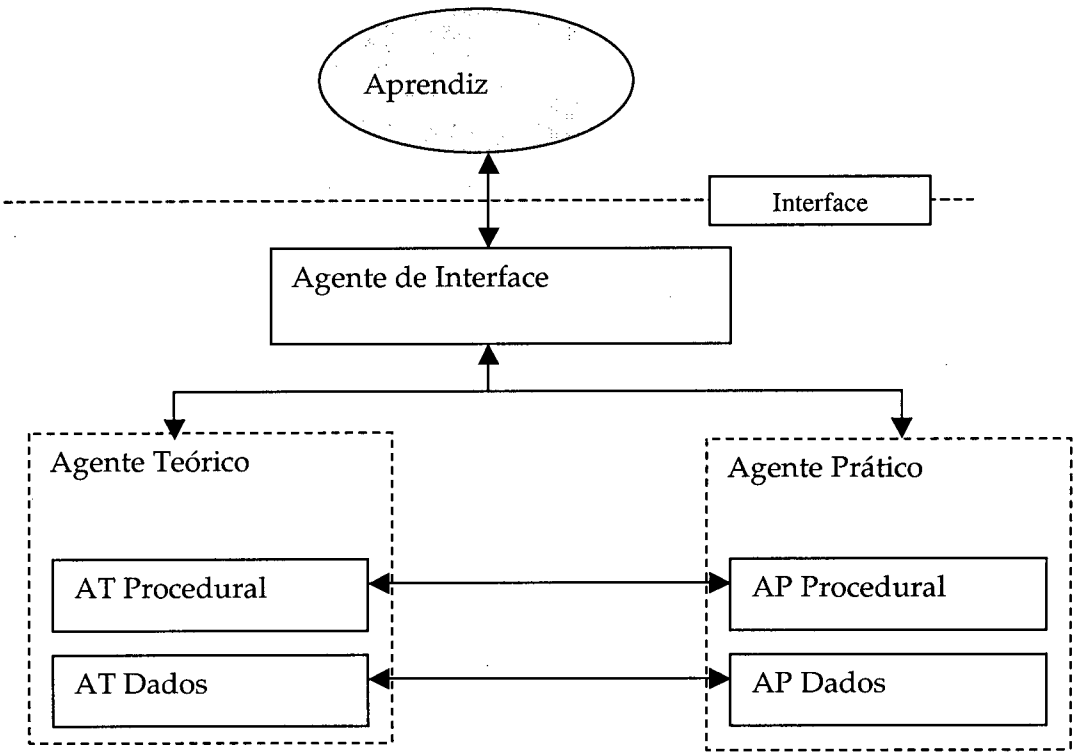

Figura 5-2 Comunicação entre os agentes do MathTutor

Os agentes conversam através da troca de mensagens KQML. Podemos verificar a seguir como são estas mensagens.

```
(ask-one
        :sender      AgenteInt
        :receiver    ATProcedural
        :language    KQML
        :content     "+id+"***"+paginaAtual+")
```

5.7 Configuração e Organização

A organização é um fator essencial no desenvolvimento de qualquer projeto, principalmente naqueles que envolvem uma ou mais equipes. Tendo em vista a continuação deste projeto e a incursão de novos desenvolvedores a Figura 5-1 mostra a configuração dos arquivos.

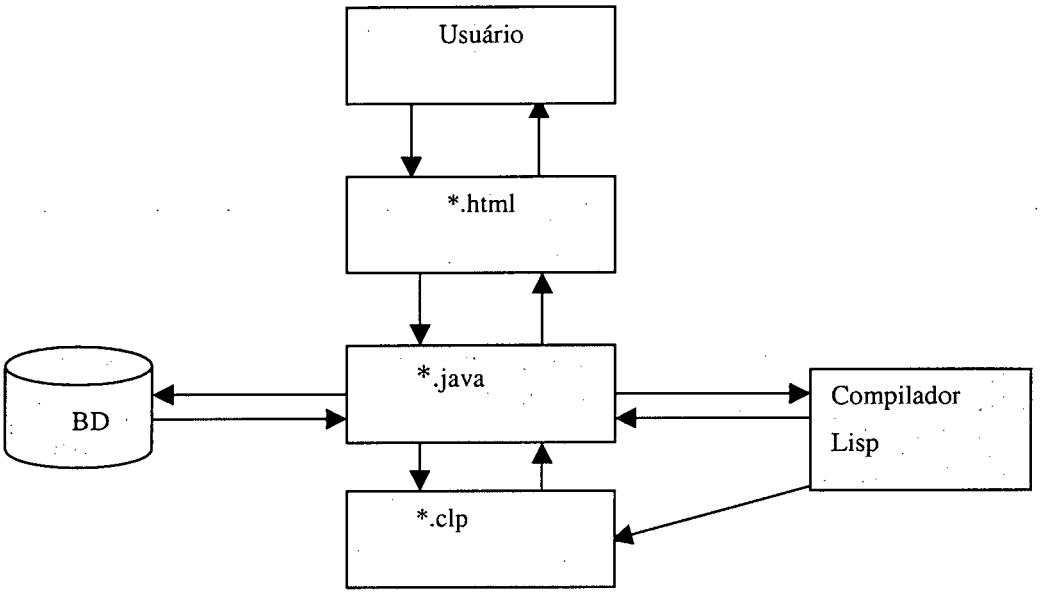

Figura 5-1 Configuração básica do mapa de arquivos do sistema

Os arquivos *.java englobam os Servlets e os arquivos que utilizam o JATLite para a comunicação entre os agentes. Já os arquivos *.clp são arquivos que formam a base de regras do sistema, algumas destas regras podem ser visualizadas no exemplo a seguir.

```
(defrule mostraResultado
    (doc agente1)
    =>
    (store "mostrar" "exercicio1"))

Regras para buscar conteúdo com a palavra Pascal
(defrule Pascal
    (preferência Pascal)
    =>
    (store "exibir1" " http://baker.das.ufsc.br:8080/tutor/servlet/... ")
    (store "titulo1" "Pascal"))
```

Na Figura 5-2 pode ser visto o que acontece internamente quando o aprendiz faz uma solicitação ao sistema. Ao clicar sobre um botão fazendo uma solicitação, o agente de interface envia uma mensagem ao agente que ele julga ser responsável por aquele determinado conteúdo. O agente ao receber a mensagem, verifica se a solicitação está dentro do seu domínio de conhecimento, caso ele não possa atender aquela solicitação repassará a mensagem para um outro agente, que deverá fazer a mesma análise.

Se o agente puder atender a solicitação do usuário, ele irá verificar na sua base de conhecimento qual a melhor atitude a ser tomada e enviará o resultado da sua inferência para apresentação ao aprendiz.

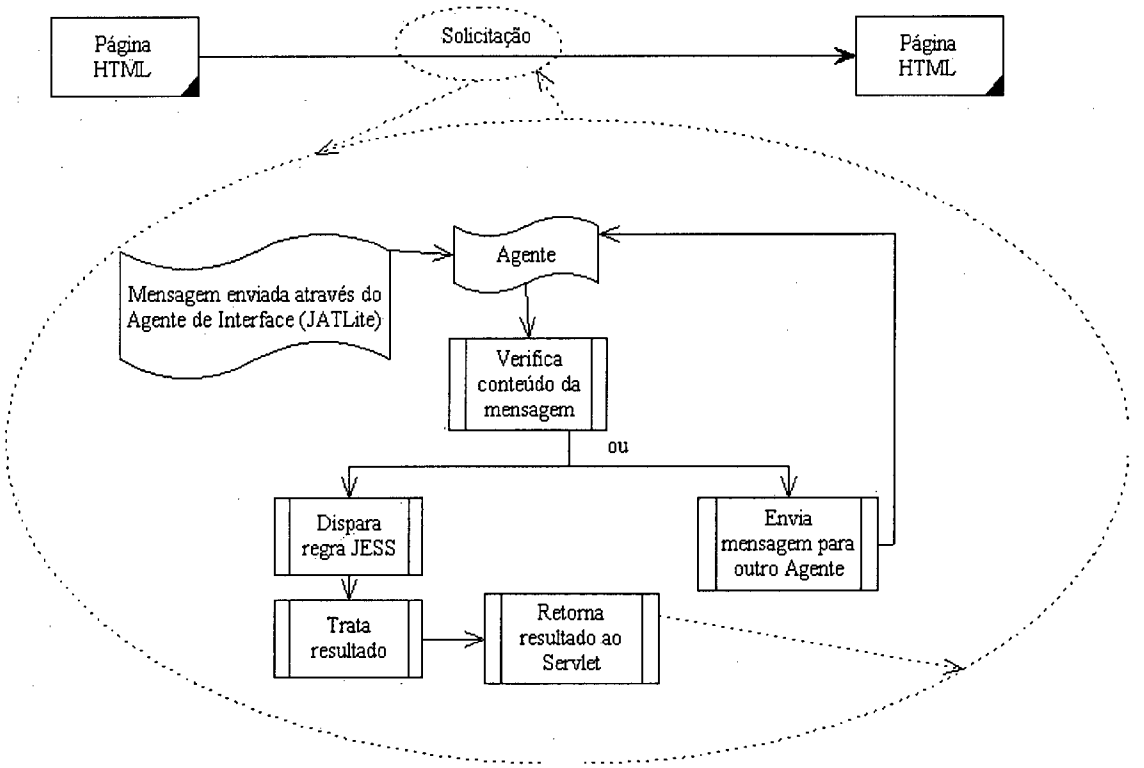

Figura 5-2 Troca de Mensagem entre os Agentes

5.8 Trabalhos Relacionados

Diversos trabalhos relacionados foram estudados objetivando a melhoria e inclusão de funcionalidades interessantes ao sistema MathTutor.

O SHART-Web [3] (Sistema tutor em HARmonia Tradicional na Web), também foi desenvolvido baseado no modelo MATHEMA. O SHART-Web trata de um Sistema Tutor Inteligente cooperativo com uma abordagem multiagente integrado a um componente hipermídia na Web e voltado para o ensino de Harmonia musical tradicional. A arquitetura e construção do sistema é bastante semelhante ao do MathTutor, embora tenham sido feitos em instituições distintas e construídos independentemente.

O AulaNet [5] é um ambiente de software baseado na Web, desenvolvido no Laboratório de Engenharia de Software - LES - do Departamento de Informática da PUC-Rio, para administração, criação, manutenção e participação em cursos à distância. Os cursos criados no ambiente AulaNet enfatizam a cooperação entre os aprendizes e entre aprendiz e docente e são apoiados em uma variedade de tecnologias disponíveis na Internet. Seu desenvolvimento de iniciou em julho de 1997.

Figura 5-1 AulaNet

Outro trabalho semelhante ao MathTutor é o ELM-ART Lisp Course [21], desenvolvido na Alemanha. É um STI na web para dar suporte ao ensino da linguagem de programação LISP. O sistema é dividido em pequenas subseções associadas com conceitos a serem estudados. Todas as interações do estudante são registradas em um modelo individual do aluno.

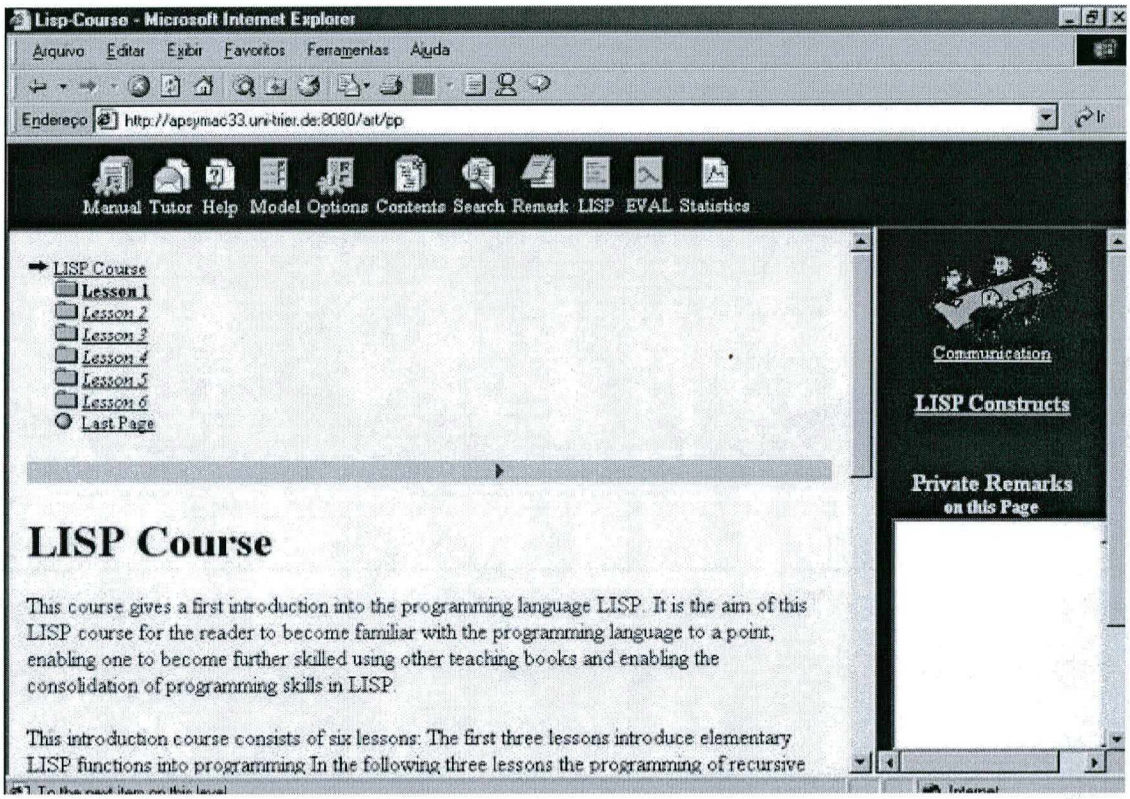

Figura 5-2 ELM-ART Lisp Course

O Eletrotutor III [7] implementa um ambiente distribuído de ensino-aprendizagem inteligente baseado em uma arquitetura multiagente. A sociedade de agentes é composta por sete agentes. Cada um dos agentes possui uma função específica. As estratégias de ensino consistem na seqüência de conteúdos, de exemplos e de exercícios que serão propostos ao aluno. A avaliação é realizada por uma série de até, no máximo, sete vezes o mesmo tipo de exercícios, para as lições que contém exercícios.

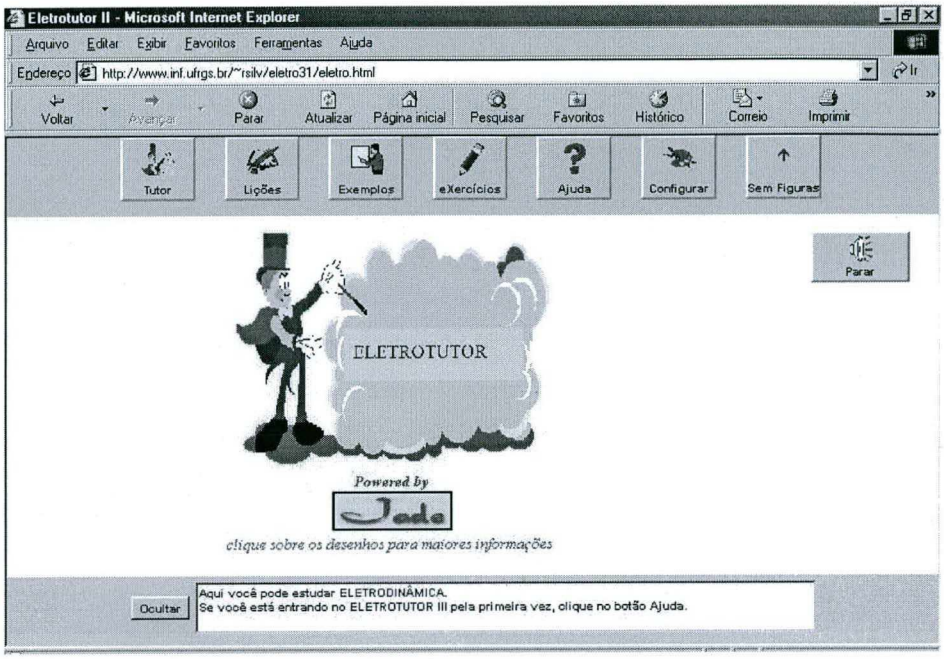

Figura 5-3 Eletrotutor

5.9 Conclusão

Neste capítulo foram apresentados os aspectos relacionados ao desenvolvimento do MathTutor.

Os assuntos aqui abordados foram:

1. Ferramentas utilizadas, com uma breve descrição sobre cada uma delas.
2. Funcionamento externo, visão do aprendiz.
3. Funcionamento interno, como, a comunicação entre os agentes e as ferramentas que fazem a operação do sistema.
4. Metas alcançadas.
5. Trabalhos Correlatos.

6 Conclusões e Perspectivas

Este trabalho foi desenvolvido com o intuito de implementar um STI Multiagente, a partir de um ambiente interativo de apoio ao processo de ensino aprendizagem. Mais de uma década passou desde o lançamento dos primeiros protótipos de STI's. Alguns exemplos de projetos de maior sucesso consumiram anos de dedicação de várias pessoas e poucos projetos saíram dos laboratórios de pesquisa. Isto mostra a complexidade deste tipo de sistema, pois projetar um STI requer uma grande compreensão das várias dimensões envolvidas no processo.

Os STI's englobam vários problemas inerentes ao campo de IA. Por exemplo, a descrição de qual informação apresentar em um ponto particular da instrução, é um problema complexo de planejamento. Podemos ainda ressaltar a incapacidade de um sistema em gerar um raciocínio pedagógico inteiramente autônomo, o que possibilitaria ao sistema tomar decisões que não tivessem sido antecipadas pelos especialistas.

A metodologia para a concretização deste trabalho partiu da pesquisa bibliográfica sobre Inteligência Artificial Distribuída, Sistemas Tutores Inteligentes e Informática na Educação. Paralelamente a isto, desenvolveu-se o estudo e familiarização com as ferramentas inerentes ao MathTutor como JAVA, JATLite, Lisp, *Servlets* e JESS.

Um aspecto a ser levado em consideração é a modelagem do aprendiz. Sabe-se que o STI deve possuir o modelo do aprendiz, de maneira a poder trabalhar individualmente de acordo com as necessidades de cada um. O MathTutor possui condições de armazenar as respostas dadas pelo aluno, assim como sua trajetória dentro do sistema, possuindo as ferramentas necessárias para se construir uma modelagem do aprendiz.

Investiu-se basicamente na concepção de uma arquitetura de um agente tutor, segundo as definições do MATHEMA, além da implementação dos agentes na SATA.

Podem-se adicionar ao sistema, além da modelagem do aprendiz, outros dois agentes responsáveis pelo domínio, o agente de manutenção que auxiliará o professor a construir um curso de maneira intuitiva, onde ele poderá inserir ou retirar um agente sem a necessidade de programar em Java ou JESS.

6.1 Contribuições

A realização deste trabalho tem como principais contribuições, concretizar um STI, seguindo o modelo MATHEMA, que é um modelo para concepção e desenvolvimento de ambientes de aprendizagem assistidos por computador, no âmbito do projeto Math_net e disponibilizar um STI que auxilie o aprendizado dos alunos do curso de Eng. de Controle e Automação Industrial (ECAI) na disciplina de Fundamentos da Estrutura da Informação (FEI).

Apêndice A

Paralelo entre noções de agentes sob o ponto de vista da Inteligência Artificial e Engenharia de Software [54]

Atualmente os "agentes de software" são o resultado da influência multidisciplinar de diferentes comunidades científicas, entre outras, das comunidades de Inteligência Artificial, Engenharia de Software e Interação Homem-Máquina.

O interesse principal da comunidade de Inteligência Artificial Distribuída consiste na solução de problemas considerados complexos que envolvam vários grupos de entidades e utilizem técnicas de IA, por exemplo redes neurais; representação simbólica de conhecimento; ou algoritmos genéticos. Os constantes avanços tecnológicos, nomeadamente a disseminação das redes de computadores, o aparecimento de máquinas paralelas potentes e, principalmente, a explosão/popularização da Internet relançaram esta área de pesquisa, ao permitir a utilização das técnicas referidas a problemas concretos, tais como: pesquisa de informação em espaços não estruturados e de enorme dimensão; processos de colaboração/negociação/interação de agentes, por exemplo, para mercados eletrônicos, distribuição de energia elétrica, controle de tráfego (aéreo ou terrestre), etc.

Por outro lado, a comunidade da Engenharia de Software (especificamente as comunidades de Linguagem de Programação e de Sistemas Distribuídos) tem vindo a preocupar-se com a modelagem e construção de aplicações distribuídas eficientes e robustas e em que a reutilização, abstração e simplicidade de programação são alguns dos aspectos preponderantes. As metáforas em geral adotadas estão associadas aos paradigmas de programação imperativa (programação estruturada e programação baseada em objetos). Os problemas colocam-se ao nível das linguagens de programação, dos sistemas operacionais e dos sistemas de bases de dados, com temas como a integração de sistemas, o acesso a base de dados, ou a interoperação entre objetos/sistemas existentes. Criou-se recentemente, no âmbito desta comunidade, um conjunto de expectativas e projetos de investigação – designados por agentes móveis – visando desenvolvimento de novas aplicações distribuídas, originalmente para o contexto de mercados eletrônicos. Técnicas como a monitoração de recursos; gestão de eventos; mecanismo de comunicação entre processos é utilizado no desenvolvimento desta classe de agentes.

Adicionalmente a comunidade de Interação Homem-Máquina também contribui e influencia o desenvolvimento de aplicações baseadas em agentes através das áreas, entre outras, de: modelagem de utilizadores; reconhecimento de imagem, voz e escrita; mundos virtuais, e de definição de interfaces multimodais [35]. Interfaces multimodais são interfaces baseadas em múltiplos meios sensoriais, tais como: conversação verbal e não verbal baseadas em linguagem natural e/ou expressões faciais [36]. O paradigma da interação social distingue-se do paradigma de interação clássica (padrão "pedido-resposta"), pelo fato da interação, para além dos mecanismos usuais de decisão e de gestão do conhecimento, basear-se em informação obtida sensorialmente (e.g., gestos, expressões faciais, sons, visão). Um exemplo significativo desta área de investigação foi desenvolvida por Nagoa e Takeuchi [42] em que um agente, dito "social", apresenta uma interface antropomórfica com rosto e capacidades de visão, audição, e fala. Outro exemplo mais recente, mas mais elementar, é o da tecnologia de agentes de software interativos da Microsoft [38].

Embora a Internet tenha tido a virtude de, ao providenciar um espaço computacional com novas características, lançar o interesse de novos projetos de investigação à volta dos agentes de software, é também, de certa forma, responsável pela confusão que o conceito de "agente" atualmente suscita. Esta confusão é principalmente devida à existência das comunidades científicas referidas, com atitudes e abordagens distintas, e à existência de inúmeras áreas de aplicação em que os atributos associados aos agentes são utilizados.

Na perspectiva da inteligência artificial existiu desde sempre a tentativa de definição do que são "agentes". Para esta comunidade, um agente é um sistema computacional, que para além das características básicas de autonomia, persistência, e sociabilidade, é também conotado com atributos geralmente associados aos seres humanos [67], [47]. É normal na inteligência artificial a caracterização de agentes usando noções mentais, tais como crença, intenções e obrigações. Outras teorias vão mais longe no processo de antropomorfismo dos agentes ao ponto de os considerarem com emoções [6].

Uma das mais compreensíveis definições de agentes é a de Wooldridge e Jennings [67] baseada numa noção "forte" e numa "fraca". As duas noções, ou visões de agentes correspondem a atributos que estes deverão possuir, os quais se dividem respectivamente em dois grupos: o de atributos essenciais e o de atributos opcionais.

A noção fraca baseia-se no conjunto mínimo de características que um agente deverá possuir, designadamente:

Autonomia – os agentes operam sem a intervenção direta dos seus ou outros utilizadores, e têm algum tipo de controle sobre as suas ações e o seu estado interno.

Sociabilidade – os agentes interatuam com outros agentes e possivelmente com os seus utilizadores.

Reatividade – os agentes analisam o seu ambiente, o qual pode ser o mundo físico; um utilizador através da sua interface gráfica; uma coleção de outros agentes; a Internet; ou talvez todos eles combinados, e respondem às alterações nele ocorridas. Estes agentes são designados por reativos. Baseiam-se sobre três componentes principais: percepção, ação e comunicação.

Pró-atividade (ou orientação por objetivos) – os agentes não atuam apenas em resposta a alterações no seu ambiente mas também apresentam comportamento conduzido por objetivos e são capazes de tomar iniciativa na realização de determinadas ações.

Persistência – os agentes mantêm o seu estado interno ao longo da sua existência.

A Figura A 1 ilustra, em termos esquemáticos, o modelo de agente, designadamente a sua relação com o ambiente que o envolve. São particularmente evidentes as capacidades de persistência (estado interno), reatividade (percepções e ações), e pró-atividade (ações). A característica de sociabilidade encontra-se esquematizada pelo bloco com a designação de "comunicação".

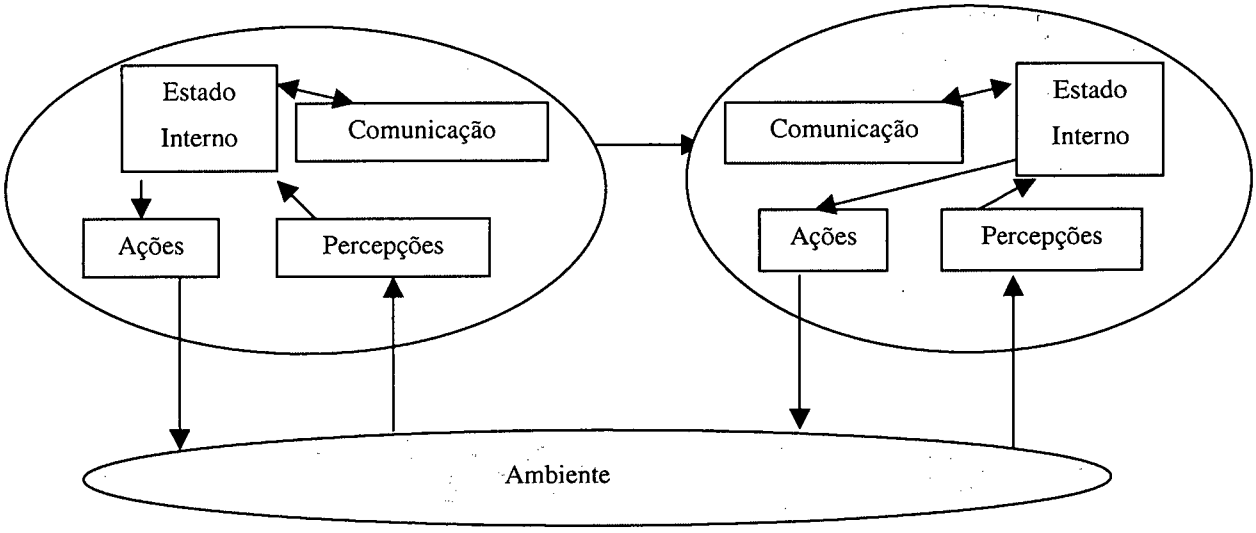

Figura A 1Modelo de Agente

Todavia, a noção de agente cognitivo, particularmente para a comunidade de inteligência artificial, apresenta um significado mais forte do que a noção (fraca) apresentada. Para esta comunidade, um agente é um sistema computacional, que para além dos atributos referidos, é também representado com atributos antropomorfos, tais como mentais ou mesmo emocionais. Adicionalmente é exigida uma representação simbólica do ambiente envolvente, capacidade cognitivas e capacidades de aprendizagem. Eis alguns dos atributos geralmente considerados adicionais à determinação de agente:

- Mobilidade – a capacidade de um agente mover-se numa rede de modo a realizar as suas tarefas e cumprir os seus objetivos. Diz-se agente móvel caso tenha capacidade de se mover, caso contrário, diz-se estático ou estacionário.
- Intencionalidade – capacidade de representação explícita dos objetivos de um agente. Estes agentes, ditos intencionais ou cognitivos, apresentam quatro componentes. Para além da percepção, ação e comunicação, apresentam ainda, capacidade de raciocínio sobre uma base de conhecimento.
- Aprendizado – a capacidade de aprendizagem está intrinsecamente associada com a capacidade de manipulação do conhecimento [2], [11]. Os agentes com capacidade de aprendizagem vão, à medida da sua utilização, reconhecendo padrões de comportamentos, padrões de preferências, etc., e atualizando a sua base de conhecimentos.
- Racionalidade – a característica de um agente (racional) não aceitar objetivos impossíveis de concretizar, contraditórios com os seus, ou ainda que não sejam, compensadores em termos do risco/custo/esforço envolvido.
- Benevolência – a suposição que um agente não tem objetivos contraditórios, e que todo o agente procura realizar as tarefas que lhe foram solicitadas. Um agente é benevolente se adota os objetivos dos outros, caso lhe seja solicitado, desde que estes não sejam incompatíveis com os seus.
- Características mentais – a definição de um modelo de agente baseado em noções mentais – conhecimento, crenças, intenções ou desejos – é uma área de atividade importante no seio da comunidade de inteligência artificial. De entre as teorias envolvidas destacam-se os trabalho de Rao e Georgeff [47] baseado em agentes BDI[3] (agentes com

[3] BDI – do inglês Beliefs, Desires and Intentions

crenças, desejos e intenções) e de Wooldrige e Jennings [67] com agentes baseados em atitudes e pró-atitudes.

Apêndice B

Modelagem do MathTutor

Para documentar o sistema, faremos uso da UML (do inglês Unified Modeling Language – Linguagem de Modelagem Unificada) para formalização das principais etapas de desenvolvimento do sistema. É importante salientar que estes diagramas servem apenas para ilustrar e melhorar o entendimento do sistema. Não tendo como objetivo abranger todos os tipos de diagramas e modelar o sistema por completo.

Caso de Uso

Este diagrama descreve as funções de cada componente dentro do sistema. É uma visão externa do sistema, apresentando como cada ator faz uso do MathTutor e o papel de cada um deles dentro do sistema.

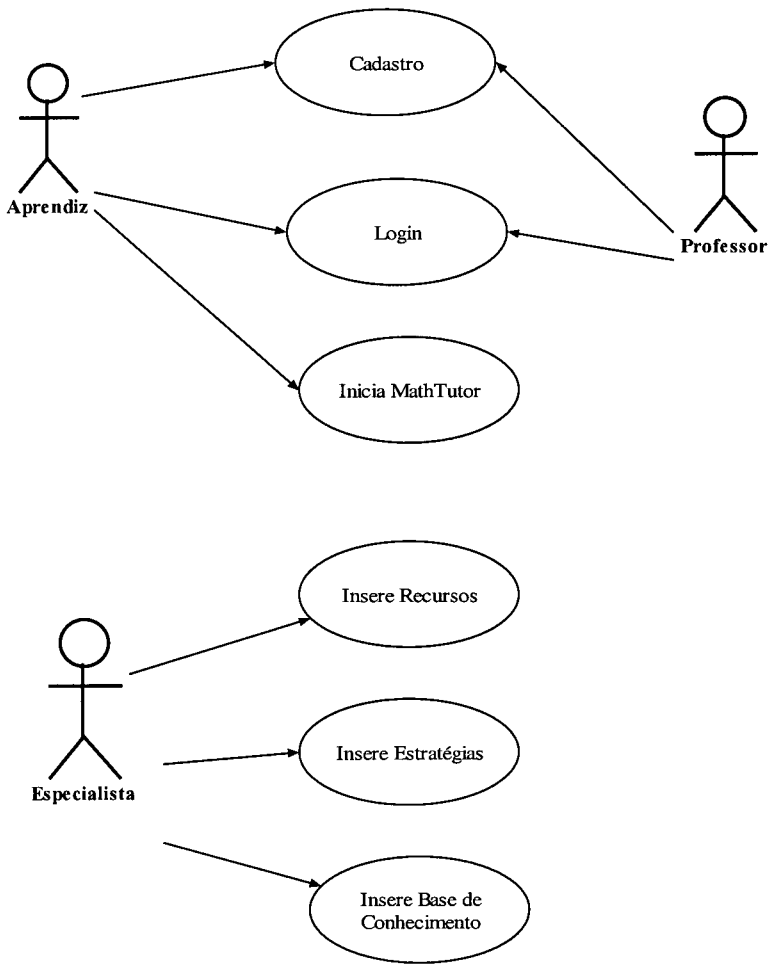

Figura B 1 Diagrama de Caso de Uso

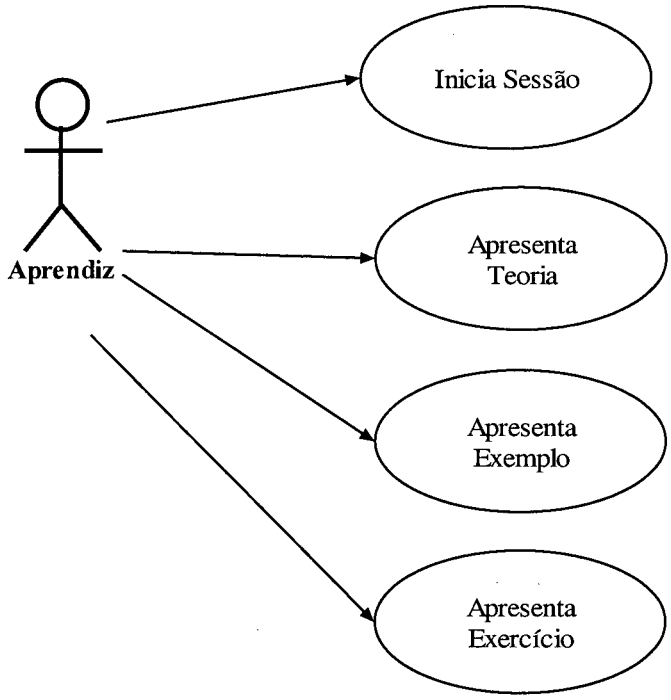

Figura B 2 Caso de Uso

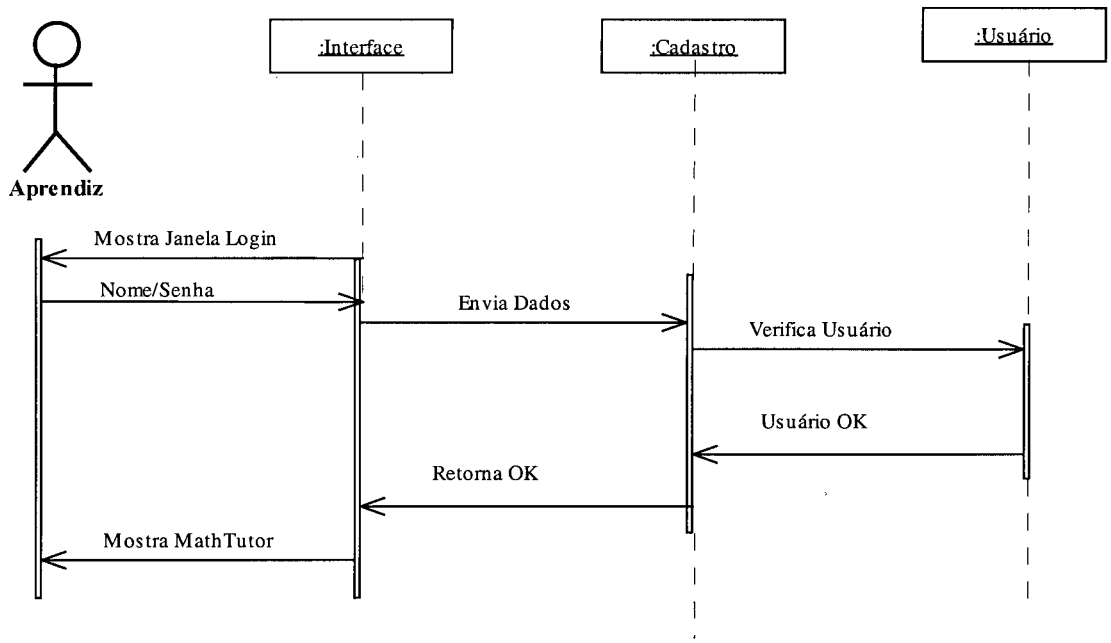

Figura B 3 Diagrama de Seqüência de Login

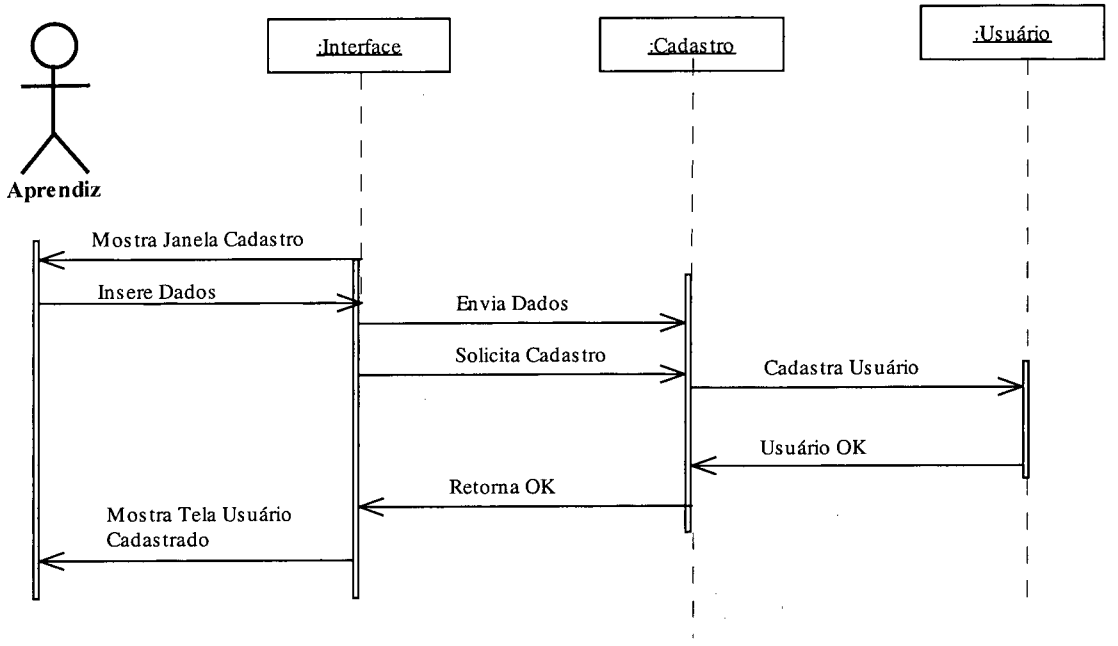

Figura B 4 Diagrama de Seqüência do Cadastro

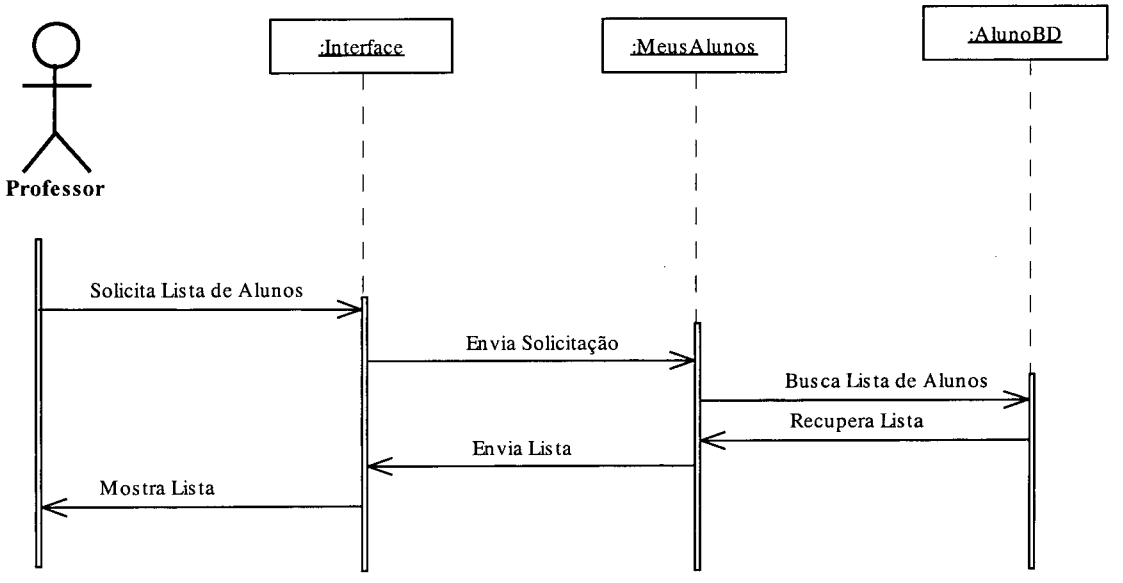

Figura B 5 Professor Solicita Alunos Cadastrados

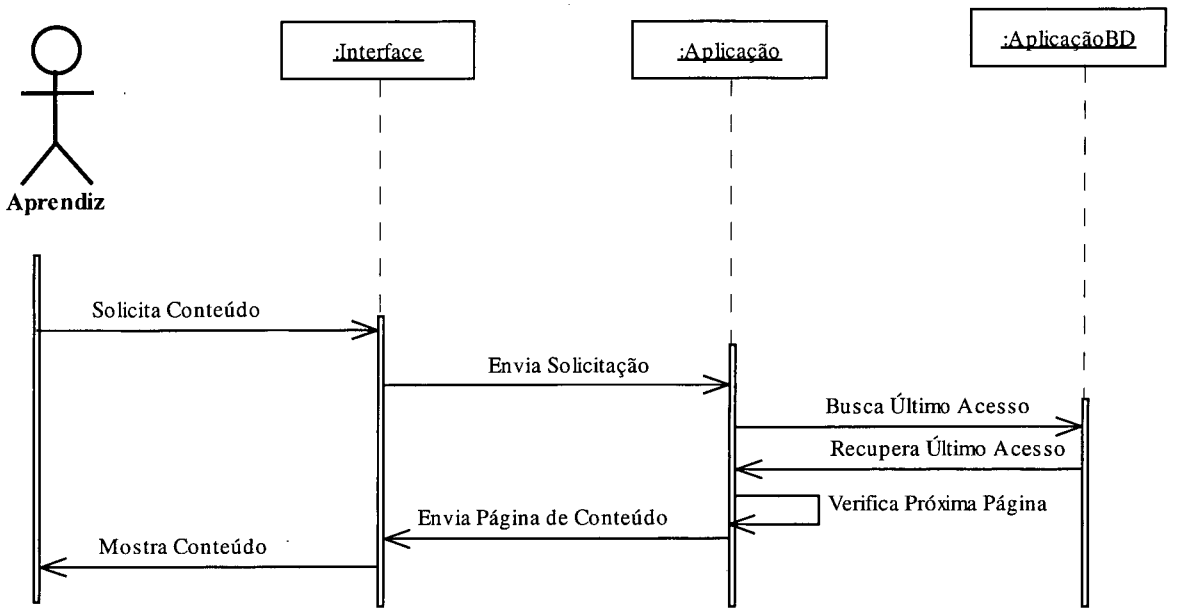

Figura B 6 Aprendiz Solicita Conteúdo

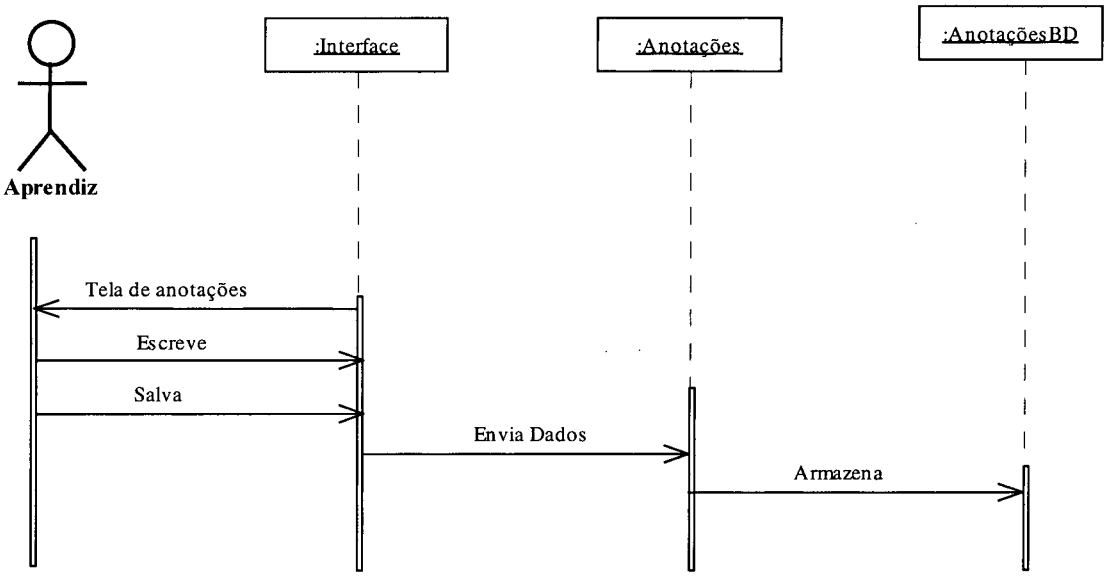

Figura B 7 Aprendiz Faz Anotações

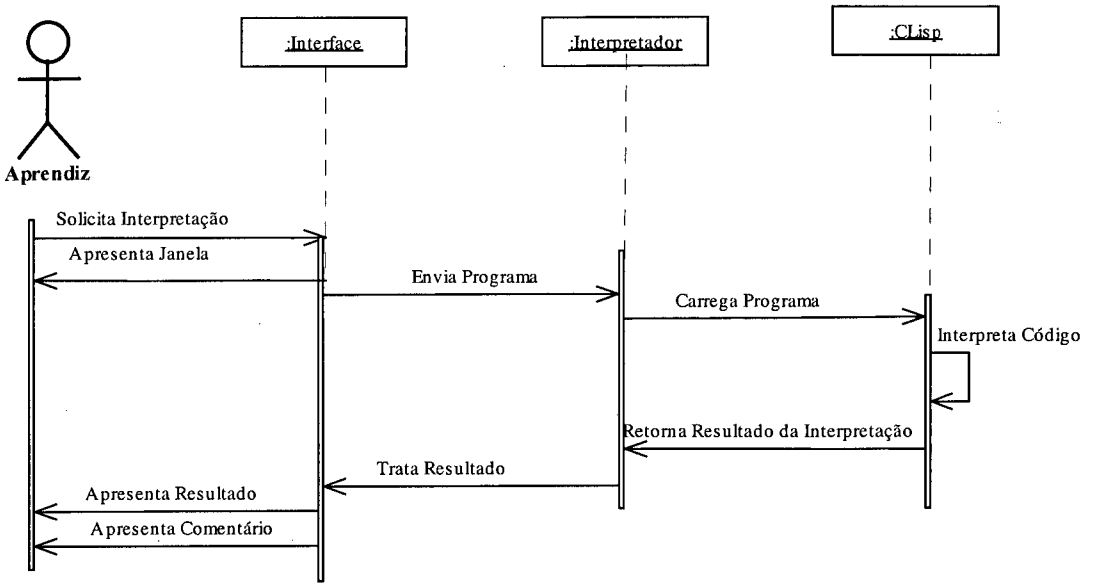

Figura B 8 Interpretação de um Exercício

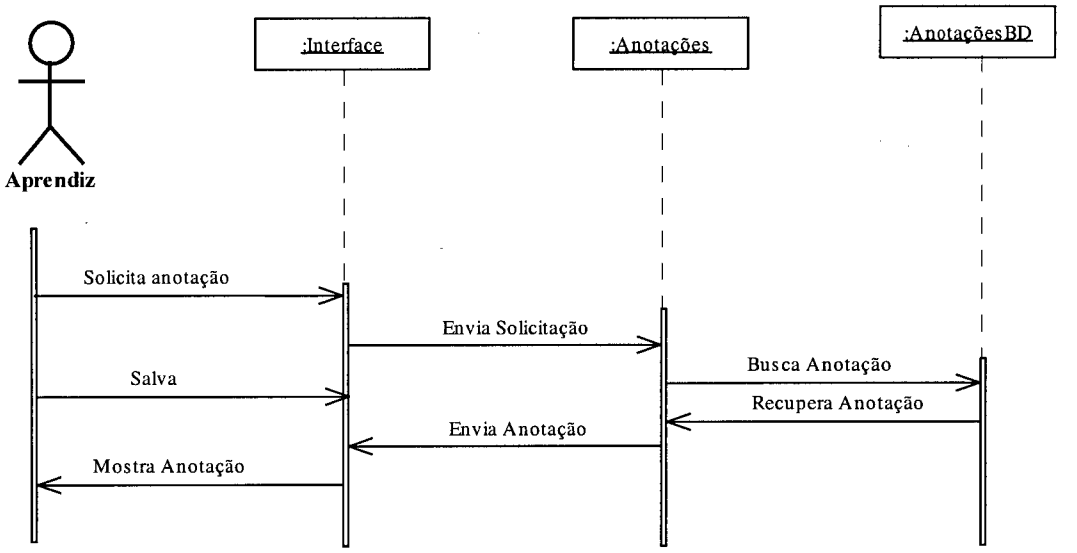

Figura B 9 Aprendiz Verifica suas Anotações

Apêndice C

Modelagem do conhecimento sobre Estrutura da Informação

A modelagem multidimensional de domínio, sugerida pelo modelo, foi aplicada ao domínio tratado pelo MathTutor. A modelagem realizada baseia-se no livro utilizado na disciplina de Fundamentos da Estrutura da Informação e também na estrutura do curso, ou seja, na distribuição das aulas, conforme se pode observar também através da página referente a esta disciplina [8]. A seguir apresenta-se, respectivamente, as visões externa e interna do domínio modelado.

C.1 Visão externa

Naturalmente, o domínio (D) do sistema consiste em Fundamentos da Estrutura da Informação. A própria distribuição atual das aulas, divididas em teóricas e práticas, sugere os contextos sob os quais o domínio pode ser analisado:

Visão Teórica (C_1) e
Visão Prática (C_2).

As profundidades relativas a tais contextos podem ser identificadas como:

Abstração Procedural (P_{11},P_{21}) e
Abstração de Dados (P_{12},P_{22}),

pois em ambos os contextos a abstração procedural é mais simples, mais concreta e mesmo historicamente anterior à abstração de dados, que a incorpora e refina. Estes contextos e profundidades definem os subdomínios que constituem D:

d_{11} = Visão Teórica da Abstração Procedural
d_{21} = Visão Prática da Abstração Procedural
d_{12} = Visão Teórica da Abstração de Dados
d_{22} = Visão Prática da Abstração de Dados

Para enfocar com sucesso os conceitos envolvidos no domínio é necessário algum conhecimento adicional, as lateralidades de acordo com o modelo MATHEMA. Este conhecimento está ligado principalmente à história da Matemática, Cálculo Numérico e à Teoria da Computabilidade, no caso do contexto teórico, e à utilização de linguagens de programação em geral e da linguagem Lisp em particular, no caso do contexto prático. Associadas à visão teórica da abstração procedural (d_{11}) podemos citar, por exemplo, os seguintes tópicos:

Ordens de crescimento (L_{111}) e
Modelo de substituição (L_{112}).

Já associadas à visão prática do mesmo contexto (d_{21}) teríamos:

Motivos para utilizar a linguagem Lisp (L_{211}) e
Avaliação de expressões pelo interpretador Lisp (L_{212}).

Como exemplos de lateralidades da visão teórica da abstração de dados (d_{12}) podem-se citar:

O significado dos dados (L_{121}) e
Encapsulamento e informação escondida (L_{122}).

Finalmente, algumas lateralidades associadas à visão prática da abstração de dados (d_{22}) consistiriam em:

Pares, listas e memória virtual (L_{221}) e
Propriedade de fechamento (L_{222}).

C.2 Visão interna

Definidos os subdomínios, estes podem ser associados a um *curriculum*, identificando as unidades pedagógicas que o constituem. Com base na experiência didática acumulada e na estrutura do livro texto, chegou-se à seguinte estrutura curricular: 7 unidades

pedagógicas associadas tanto à visão teórica (d_{11}) quanto à visão prática (d_{21}) da abstração procedural e 8 unidades pedagógicas associadas às visões teórica (d_{12}) e prática (d_{22}) da abstração de dados.

As 7 unidades pedagógicas associadas a d_{11} e d_{21} são:

Funções Primitivas,
Funções Compostas,
Iteração e Recursão,
Funções como Objetos Manipuláveis,
Funções como Argumentos,
Funções como Métodos Gerais e
Funções como Resultados de Funções.

Já as 8 unidades pedagógicas associadas a d_{12} e d_{22} são:

Dados Compostos,
Barreiras de Abstração,
Dados Hierárquicos,
Seqüências como Interfaces Convencionais,
Dados Simbólicos,
Múltiplas Representações para Dados Abstratos,
Programação Direcionada a Dados e Aditividade e
Sistemas com Operações Genéricas.

As unidades pedagógicas formam o conteúdo do curso MathTutor, onde cada uma das unidades compõem uma lição do curso.

Apêndice D

CÓDIGO DO AGENTE TEÓRICO PROCEDURAL

```
import java.io.*;
import java.util.*;
import jess.*;
import Abstract.*;
import KQMLLayer.*;
import RouterLayer.*;
import RouterLayer.AgentClient.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class ATProcedural extends RouterClientAction {

/**
* The return value, as a result of arithmetic. Example of a data
member
*/
   String _returnValue;
   //TProcedural ATaux = new TProcedural();
   private Rete motor = new Rete();
   public String paginaRetorno;
   String paginaMostrar = "";
   int tam, id;
   String mostrar;
   public String Valor;
   int DOC = 1;
   int DOCMAX = 1;
   public String textoAtual = "";
   public String tipo1 = "";
   public String retorno  = "";
   public String botao1 = "";
   public String botao2 = "";
   public String tipo2 = "";
   public String texto1 = "";
   public String texto2 = "";
      /**
* Will be used by dynamic class downloading. The RCAddressHelper will
use
* this constructor. After construction, setMyAddress(Address) and
* setRouterAddress(Address) should be invoked before invoke connect
method.
*/
   public ATProcedural()  {
      super();
      try{
         motor.executeCommand("(load-facts /var/httpd/jakarta-
tomcat/webapps/tutor/WEB-INF/classes/MathTutor/BaseDeDados.clp)");
         motor.executeCommand("(batch /var/httpd/jakarta-
tomcat/webapps/tutor/WEB-INF/classes/MathTutor/Query.clp)");
      }
         catch(JessException e) {
            e.printStackTrace();
         }
   }
```

```
   /**
   * Router client constructor. Since the agent is stand alone, it will
   * pass its address and router address. Maximum idle time can be set,
   * by passing the durationtime parameter.If the registerrequest is
true,
   * it will try to register to the router using registraraddress.
   * @param myaddress My address
   * @param routeraddress Router address
   * @param registraraddress Registrar address
   * @param durationtime Maximum idle time for the receiver thread
   * @param registerrequest If true, registration will be performed
   */
      public ATProcedural(Address myaddress, Address routeraddress,
Address registraraddress, int durationtime) {
      // use RouterLayer.AgentClient.RouterClientAction constructor

         super(myaddress,routeraddress,registraraddress,durationtime);
         try{
            motor.executeCommand("(load-facts /var/httpd/jakarta-
tomcat/webapps/tutor/WEB-INF/classes/MathTutor/BaseDeDados.clp)");
            motor.executeCommand("(batch /var/httpd/jakarta-
tomcat/webapps/tutor/WEB-INF/classes/MathTutor/Query.clp)");
         }
            catch(JessException e) {
               e.printStackTrace();
            }
      }



      public boolean Act(Object o)  {
         try  {
            String stampmsg = (String)o;
            String content = null;
            System.out.println("PASSEI POR AQUI 1");
         //KQMLmail wrap the KQMLmessage
            KQMLmail mail = new KQMLmail(stampmsg,0);

         // add the mail to the _mailQueue.
         // this step is essential to delete message after execution
            _mailQueue.addElement(mail);

         // use KQMLmail getKQMLmessage() to get KQMLmessage object
            KQMLmessage kqml = mail.getKQMLmessage();

         // this is a typical way to find out the performative
            String perf = kqml.getValue("performative");

            if(perf.equals("ask-one")){
               content = kqml.getValue("content");
               String receiver = kqml.getValue("sender");
            }
            if(content == null)  {
               sendErrorMessage(kqml);
               return false;
            }
```

```
//  Novo tratamento da mensagem (Dividida com o StringTokenizer)
            StringTokenizer token = new StringTokenizer(content, "***");
            id = Integer.parseInt(token.nextToken());
            paginaMostrar = token.nextToken();
            setPronto(false);
            if (token.hasMoreTokens()) {
               paginaRetorno = token.nextToken();
               setPaginaRetorno(paginaRetorno);
            }
            if(perf.equals("ask-one"))  {
               addToDeleteBuffer(0);

               if (SQLManager.getDefaultSQLManager().getAgente(id)) {
                  System.out.println("getAgente true id="+id);
                  motor.executeCommand("(batch /var/httpd/jakarta-
tomcat/webapps/tutor/WEB-INF/classes/MathTutor/Query.clp)");
                  SQLManager.getDefaultSQLManager().setAgente(id, false);
               }


               motor.store("mostrar", "FIM");
               motor.executeCommand("(next "+paginaMostrar+")");
               Value resultado = motor.fetch("mostrar");
               Valor = resultado.stringValue(motor.getGlobalContext());
               setProxima(Valor);
               System.out.println("resultado=" +resultado);
               System.out.println("Valor=" +Valor);

               if (Valor.equals("agente2")) {
               /** Nova Mensagem: */
                  String msg = "(ask-one :sender ATProcedural :receiver
APProcedural :language KQML :content "+id+"***"+Valor+")";
                  sendKQMLMessage(msg);
                  System.out.println("ATProcedural -> APProcedural");
               }
               else {
                  setPronto(true);
                  String acesso = Valor;
                  System.out.println("acesso= " +acesso);
               }
               System.out.println("Valor=" +Valor);
               System.out.println("DOC=1=" +DOC);
            }


            catch (JessException e) {
               e.printStackTrace();
            }
            catch (KQMLLayer.ParseException e)  {
               e.printStackTrace();
            }
            catch (ConnectionException e)  {
               e.printStackTrace();
            }
            catch (SQLException e) {
               e.printStackTrace();
            }
         return true;
      }
```

```
      public void getValues(int id) {
         try {
            String proxima=getProxima(id);
            System.out.println("Proxima :"+proxima);
            motor.executeCommand("(grava "+ proxima +")");
            Value resultado1 = motor.fetch("jessTextoAtual");
            textoAtual =
resultado1.stringValue(motor.getGlobalContext());
            Value resultado2 = motor.fetch("jessTipo1");
            tipo1 = resultado2.stringValue(motor.getGlobalContext());
            Value resultado3 = motor.fetch("jessRetorno");
            retorno = resultado3.stringValue(motor.getGlobalContext());
            Value resultado4 = motor.fetch("jessBotao1");
            botao1 = resultado4.stringValue(motor.getGlobalContext());
            Value resultado5 = motor.fetch("jessBotao2");
            botao2 = resultado5.stringValue(motor.getGlobalContext());
            Value resultado6 = motor.fetch("jessTipo2");
            tipo2 = resultado6.stringValue(motor.getGlobalContext());
            Value resultado7 = motor.fetch("jessTexto1");
            texto1 = resultado7.stringValue(motor.getGlobalContext());
            Value resultado8 = motor.fetch("jessTexto2");
            texto2 = resultado8.stringValue(motor.getGlobalContext());

         }
            catch(JessException e) {
               System.out.println(e);
            }
      }


   /**
   * Send error message of the received message
   * @param kqml KQMLmessage received
   */
      protected void sendErrorMessage(KQMLmessage kqml)  {
         try {
            String receiver = kqml.getValue("sender");
            String sendmsg = "(error :sender ";
            sendmsg = sendmsg + getName() + " :receiver " + receiver;
         //use getSendString() method of the KQMLmessage object
            sendmsg = sendmsg + " :language KMQL :content (" +
kqml.getSendString() + "))";
            sendMessage(sendmsg);
         }
            catch (ConnectionException c)  {
               System.err.println(c);
            }
            catch (KQMLLayer.ParseException p) {
               System.err.println(p);
            }
      // should be deleted after execution. Send delete-message message
to the Router
         addToDeleteBuffer(0);
      }


      // does not implement processMessage
      public void processMessage(String s,Object obj) {
      }
```

```
    /**
    * Main program
    */

       public void inicia(String end) {
          try  {
             Address myAddress = null;
             Address routerAddress = null;
             Address registrarAddress = null;
             boolean registerRequest = false;
             int idletime = 1000;
             BufferedReader in = new BufferedReader(new FileReader(new
File(end)));
             String line;
             while((line=in.readLine()) != null)  {
                String next;
                if(line.startsWith("MyAddress"))  {
                   next = in.readLine();
                   myAddress = new Address(next);
                   setMyAddress(myAddress);
                }
                else if(line.startsWith("RouterAddress"))  {
                   next = in.readLine();
                   routerAddress = new Address(next);
                   setRouterAddress(routerAddress);
                }
                else if(line.startsWith("RouterRegistrarAddress"))  {
                   next = in.readLine();
                   registrarAddress = new Address(next);
                   setRegistrarAddress(registrarAddress);
                }
                else if(line.startsWith("RegisterRequest"))  {
                   next = in.readLine();
                   registerRequest = next.startsWith("y");
                }
                else if(line.startsWith("MaxIdleTime"))  {
                   next = in.readLine();
                   idletime = (Integer.valueOf(next)).intValue();
                }
             }
             in.close();
             ATProcedural server = new ATProcedural(myAddress,
routerAddress, registrarAddress, idletime);
          // Since stand alone, create server thread using my address

server.createServerThread(myAddress.getID(),Thread.NORM_PRIORITY);
             if(registerRequest == true)
server.register(registrarAddress,myAddress);
          // connect to the router by sending my address
             server.connect(myAddress);
          // start to receive messages *****
             server.start();
          //sendResult(rec);

          }
             catch (IOException e)  {
                e.printStackTrace();
             }
             catch (ConnectionException e) {
```

```
                e.printStackTrace();
            }

        }


    //Database
        public String getProxima(int id) {
            String result = "texto1";
            try {
                ResultSet rs =
SQLManager.getDefaultSQLManager().executeQuery("select ultimapagina from
aplicacao where id_usuario="+id);
                if(rs.next())
                    result = rs.getString("ultimapagina");
            }
                catch (SQLException e) {
                    System.err.println("Erro pegando proxima : "+e);
                }
                finally {
                    return result;
                }
        }

        public String getPaginaRetorno(int id) {
            String result = "texto1";
            try {
                ResultSet rs =
SQLManager.getDefaultSQLManager().executeQuery("select paginaretorno from
aplicacao where id_usuario="+id);
                if(rs.next())
                    result = rs.getString("paginaretorno");
            }
                catch (SQLException e) {
                    System.err.println("Erro pegando retorno : "+e);
                }
                finally {
                    return result;
                }
        }

        private void setProxima(String prox) {
            try {
                ResultSet rs =
SQLManager.getDefaultSQLManager().executeQuery("select * from aplicacao
where id_usuario="+id);
                if (rs.next())
                    SQLManager.getDefaultSQLManager().executeUpdate("update
aplicacao set ultimapagina=\'"+prox+"\' where id_usuario="+id);
                else
                    SQLManager.getDefaultSQLManager().executeUpdate("insert
into aplicacao(id_usuario, ultimapagina) values("+id+", '"+prox+"\')");
            }
                catch (SQLException e) {
                    System.err.println("Erro gravando proxima : "+e);
                }
        }

        private void setPaginaRetorno(String ret) {
```

```
         try {
            ResultSet rs =
SQLManager.getDefaultSQLManager().executeQuery("select paginaretorno from
aplicacao where id_usuario="+id);
            if (rs.next())
               SQLManager.getDefaultSQLManager().executeUpdate("update
aplicacao set paginaretorno=\'"+ret+"\' where id_usuario="+id);
            else
               SQLManager.getDefaultSQLManager().executeUpdate("insert
into aplicacao(id_usuario, paginaretorno) values("+id+", '"+ret+"\')");
         }
            catch (SQLException e) {
               System.err.println("Erro gravando retorno : "+e);
            }
      }

      private void setPronto(boolean pto) {
         try {
            ResultSet rs =
SQLManager.getDefaultSQLManager().executeQuery("select pronto from
aplicacao where id_usuario="+id);
            if (rs.next())
               SQLManager.getDefaultSQLManager().executeUpdate("update
aplicacao set pronto=\'"+pto+"\' where id_usuario="+id);
            else
               SQLManager.getDefaultSQLManager().executeUpdate("insert
into aplicacao(id_usuario, pronto) values("+id+", "+pto+")");
         }
            catch (SQLException e) {
               System.err.println("Erro gravando pronto : "+e);
            }
      }

      public boolean getPronto(int id) {
         boolean result = true;
         try {
            ResultSet rs =
SQLManager.getDefaultSQLManager().executeQuery("select pronto from
aplicacao where id_usuario="+id);
            if(rs.next())
               result = rs.getBoolean("pronto");
         }
            catch (SQLException e) {
               System.err.println("Erro pegando pronto : "+e);
            }
            finally {
               return result;
            }
      }


   }
```

Referências Bibliográficas

[1] APACHE. Disponível em: <http://java.apache.org/>, Acesso em: 16 junho 2001.

[2] ARMSTRONG, R. et alli. "WebWatcher: A Learning Apprentice for the WWW". *Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments*, 1995.

[3] ASSUNÇÃO, F. M. "Shart-Web: Um Sistema Tutor de Harmonia Tradicional na Web". Universidade Federal de Paraíba - Dissertação de Mestrado, 2001 (Mestrado em Engenharia Elétrica).

[4] AUBERGER, M. "Student Modeling in Educational Multimedia Titles Using Agents". Disponível em: <http://aif.wu-wien.ac.at/usr/geyers/>, Acesso em: 18 maio 1998.

[5] AULANET. Disponível em: <http://guiaaulanet.eduweb.com.br/index.asp>, Acesso em: 18 março 2001.

[6] BATES, J. "The role of emotion in believable agents". *Communications of the ACM*, 37(7), Julho, 1994.

[7] BICA, F. "Eletrotutor III - Uma Abordagem Multiagente para o Ensino à Distância". Porto Alegre: UFRGS, 2000. Dissertação de Mestrado.

[8] BITTENCOURT, G. "Fundamentos da Estrutura da Informação", Disponível em: <http://das.ufsc.br/~gb/fei>, Acesso em: 09 janeiro 2002.

[9] _____ *Inteligência Artificial: Ferramentas e Teorias*, capítulo 5, Editora da UFSC, 1998.

[10] BRATMAN, M. "What is Intention?" *In: Cohen, P; Morgan, J; Pollack, M.* (Eds.), 1989. Anais: MIT Press, 1989.

[11] CAMARINHA – MATOS, L.M., Vieira, W. "Using Multiagent systems and the Internet in care services for the agent society." *Proc. Of BASYS'98 – 3º IEEE/IFIP Int. Conf. On Balanced Automation Systems, Intelligent Systems for Manufacturing* (Kluwer Academic), pp. 33-47, ISBN 0-412-84670-5, Prague, Czech Republic, Aug 98.

[12] CLANSEY, W.J., *Knowledge-Based Tutoring: The GUIDON Program*, The MIT Press, 1987

[13] CLISP (Common Lisp). Disponível em: <http://clisp.sourceforge.net/>, Acesso em: 25 janeiro 2001.

[14] CORRÊA, M. "A arquitetura de diálogos entre agentes cognitivos distribuídos" *Tese de Doutorado*, Rio de Janeiro: UFRJ, 1994.

[15] COSTA, E. de B. "Um Modelo de Ambiente Interativo de Aprendizagem Baseado numa Arquitetura Multiagentes", *Tese de Doutorado*, Universidade Federal da Paraíba, Campina Grande, 1997. Tese (Doutorado em Engenharia Elétrica).

[16] DAVIDSSON, Paul "Concept Acquisition by Autonomous Agents: Cognitive Modeling versus the Engineering Approach". Lund University Cognitive Studies 12, ISSN 1101-8453, Lund University: Sweden, 1992

[17] DEMAZEAU, Y.; MÜLLER, J. P. "Decentralized Artificial Intelligence". *Elsevier Science Publishers,* North-Holland, 1990.

[18] DROGOUL, A.; FERBER, J." Using reactive multi-agent systems in simulation and problem solving. Distributed Artificial Intelligence: Theory and Praxis", *Distributed Artificial Intelligence: Theory and Praxis,* 1992

[19] DURFEE, E. ; ROSENSCHEIN, J. "Distributed Problem Solving and Multi-Agent Systems : Comparisons and Muti-Agent Systems : Comparison and Examples".*13th International Workshop on Distributed Artificial Intelligence*, Washington, July 17-19, 1994.

[20] EKDHAUL, et alli "Towards Anticipatory Agents." M. Woolridge and N. R. Jennings, editors, *Intelligent Agents Theories, Architectures, and Languages.* Lecture Notes in Artificial Intelligence 890, pp. 191-202, Springer Verlag, 1995.

[21] ELM-ART LISP COURSE, Disponível em: <http://apsymac33.uni-trier.de:8080/elm-art/login-e>, Acesso em:25 janeiro 2001.

[22] FARIA, T. de F. ; BITTENCOURT, G. "Um ambiente interativo Multiagentes para o ensino de estrutura da informação", *XI Simpósio Brasileiro de Informática na Educação (SBIE'2000)*, nov 2000.

[23] FARIA, T. F. "Um Ambiente Interativo Multiagentes para o Ensino de Estrutura da Informação" Universidade Federal de Santa Catarina - *Dissertação de Mestrado*, 2001,. (Mestrado em Engenharia Elétrica).

[24] FRANKLIN, S. ; GRAESSER, A. "Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents" Intelligent Agents {III}. *Agent Theories, Architectures and Languages* ({ATAL}'96), Springer-Verlag.

[25] FRIEDMAN-HILL, E. J. "Jess, The Java Expert System Shell", Sandia National Laboratories, 1999, Disponível em: <http://herzberg.ca.sandia.gov/jess>, Acesso em: 25 janeiro 2001.

[26] GIRAFFA, L. M. M. "Comparações entre sistemas de ensino inteligente." Disponível em: < http://www.inf.pucrs.br/~giraffa>. Acesso em: 25 maio 2001.

[27] _____ "Uma arquitetura de tutor utilizando estados mentais" *Tese de Doutorado*, Porto Alegre: CPGCC/UFRGS, 1999..

[28] GREER, J. E. ; McCALLA, G. I. "Student Modelling: The Key to Individualized Knowledge-Based Instruction", Capítulo 1, NATO ASI Series, vol. 125, 1991.

[29] HIX, D. ; HARTSON, H.R. "Developing User Interfaces", John Wiley & Sons, Inc.,1993.

[30] Disponível em: <http://penta2.ufrgs.br/edu/teleduc/teledwbt.htm>. Acesso em: 10 maio 2002.

[31] JAKARTA TOMCAT. Disponível em: <http://jakarta.apache.org/tomcat/>. Acesso em: 10/12/00.

[32] JATLITE. "Java Agent Template Lite." , Stanford University, 1997. Disponível em: <http://Java.stanford.edu/Java_agent/html/index2.html>. Acesso em: 10 maio 2002.

[33] JAVA(tm) SERVLET TECHNOLOGY. Disponível em: <http://java.sun.com/products/servlet/index.html>. Acesso em: 01 dezembro 2000.

[34] JONASSEN, D.H.; WANG, S. "The Physics Tutor: Integrating Hypertext and Expert Systems", Journal of Educational Technology Systems, Vol. 22(1), pp. 19-28, 1993.

[35] KALAWSKY, R.. *The Science of Virtual Reality and virtual Environments.* (2ª edição), Addison-Wesley, 1998

[36] MAES, P. "Artificial Intelligence meets Entertainment: Lifelike Autonomous Agents", *Communications of the ACM* 38 (11), pp. 108-114,Nov. 1995.

[37] MATH_NET, Equipe. Projeto Math_net "Uma abordagem via sistemas Multiagentes para concepção e realização de ambientes interativos de aprendizagem cooperativa assistidos por computador". Disponível em: *<http://www.lcmi.ufsc.br/mathnet/>*. Acesso em: 10 outubro 2001.

[38] MICROSOFT. *ActiveX Technology for Interactive Software Agents*,1996. Disponível em: <http://www.microsoft.com/intdev/agent/agent.htm>. Acessado em: 15 maio 2001.

[39] MINSKY, M. "A framework to represent knowledge. In *The Psychology of Computer Vision"*, pages 211-277. McGraw-Hill, 1975.

[40] MINSKY, M. *The Society of Mind*, Simon and Schuster, New York, 1985.

[41] MURRAY, T. "Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art". *Journal of Artificial Intelligence and Education*, vol. 10, 1999.

[42] NAGOA, K; Takeuchi, A. "Social Interaction: Multimodal Conversation with Social Agents", *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, Seattle, WA (August 1-4, 1994), Vol. 1, pp. 22-28. 1995.

[43] NWANA, H. "Software Agents : An Overview", *Knowledge Engineering Review*, vol. 11, N.3, 1996.

[44] PARK, O., "Functional Characteristics of Intelligent Computer-Assisted Instruction: Intelligent Features", *Educational Technology*, June 1988, pp. 7-14.

[45] PIAGET, J., *A. Equilibração das Estruturas Cognitivas - Problema Central do Desenvolvimento.* Rio de Janeiro: Zahar Editores, 1976. Original: L'équilibration des structures cognitives - Probléme central du développment. Paris: Presses Universitaires de France, 1975.

[46] POSTGRESQL. Disponível em: <www.postgresql.org/>. Acesso em: 05 julho 2001.

[47] RAO, P. G. "BDI Agents: from Theory to Practice". *Proceedings of the First International Conference on Multi-Agent Systems*, 1995.

[48] RICH, E. *Artificial Intelligence*, McGraw-Hill, Inc., 1991.

[49] SANTOS, N. "Computadores na Educação: discutindo alguns pontos críticos." *Em aberto*, ano 12, n. 57, p. 27-31, Brasília, 1993.

[50] SEARLE, J. R. *Expression and Meaning: Studies in the Theory of Speech Acts*, tradução : Ana Cecilia G. A. de Camargo, Ana Luiza Marcondes Garcia.- São Paulo, Martins Fontes, 1995.

[51] SHERER, R. ; SCHLAGETER, G., *Multi-Agent Approach for Integration of Neural Networks and Expert Systems*, John Wiley & Sons, 1995.

[52] SHNEIDERMAN, B., *Designing The User Interfaces: Strategies for Effective Human Computer Interaction*, Addisson-Wesley, 1992.

[53] SHORTLIFFE, E.H., "Computer-Based Medical Consultations: MYCIN", New York: *American Elsevier*, 1976.

[54] SILVA, A. *Agentes de Software na Internet*, Edições Centro Atlântico, Portugal, 1999.

[55] STEELE Jr., GUY L., *Common Lisp - The Language*. Digital Press. 2 edição, 1990.

[56] STEFIK, M. *Introduction to Knowledge Systems*. Morgan Kaufman, Inc. 1995.

[57] TROLLIP, S.R. & ALESSI, S.M., *Computer - Based Instruction , Methods and Development*. Second Edition, Prentice Hall, INC, 1991.

[58] TURING, A. M. "Computing Machinery and Intelligence". *Mind*, 54(236): 433-460, October 1950.

[59] UFSC-TEAM. Disponível em: <http://www.das.ufsc.br/ufsc-team>. Acesso em: 05 junho 2001.

[60] VALENTE, J. A. "Informática na educação: conformar ou transformar a escola: Perspectiva". Núcleo de Publicações – CED/UFSC, ano 13, n.24,p. 41-49, Florianópolis, Julho-Dezembro, 1995.

[61] VICCARI, R et alli. "BDI models and systems: reducing the gap" In: *Agents Theory, Architecture and Languages Workshop*, 1998, Canarias. Proceedings : Springer-Verlag, 1998.

[62] VICCARI, R.M., "Inteligência Artificial e Educação-Indagações Básicas", *Anais do IV Simpósio Brasileiro de Informática na Educação*, Recife, 1993.

[63] WATERMAN, D.A. *A Guide to Expert Systems*, Addison-Wesley Publishing Company, Reading, MA, 1986.

[64] WEISS, G. *Multiagent Systems : Modern Approach to Distributed Artificial Intelligence,* MIT Press, 1999.

[65] WENGER, E. *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann Publishers, Inc., 1987.

[66] WOOLDRIDGE, M & JENNINGS, N. R. , "Agent Theories, Architectures, and Languages: a Survey," em Wooldridge and Jennings Eds., *Intelligent Agents*, Berlin:Springer-Verlag, 1-22, 1995.

[67] WOOLDRIDGE, M.; JENNINGS, N. R.: "Intelligent Agents: Theory and Practice", *Knowledge Engineering Review* Volume 10 No 2, pp.115-152, June 1995.

[68] ZAMBERLAM, A O; GIRAFFA, L. M. M.; MORA, M. C. "Um editor para programação orientada á agentes BDI" In*: II Workshop sobre Ambientes de Aprendizagem Baseados em Agentes XI SBIE 2000*, Maceió. XI SBIE 2000 publicação do resumo do workshop. Maceió: SBC/UFAL, 2000. p. 481-482.

[67] WOOLDRIDGE, M.; JENNINGS, N. R.: "Intelligent Agents: Theory and Practice", *Knowledge Engineering Review* Volume 10 No 2, pp.115-152, June 1995.

[68] ZAMBERLAM, A O; GIRAFFA, L. M. M.; MORA, M. C. "Um editor para programação orientada á agentes BDI" In*: II Workshop sobre Ambientes de Aprendizagem Baseados em Agentes XI SBIE 2000*, Maceió. XI SBIE 2000 publicação do resumo do workshop. Maceió: SBC/UFAL, 2000. p. 481-482.